MASTER EN SCIENCES INFORMATIQUES

Problèmes de sécurité dans un environnement "temps réel"
Étude critique d'un système particulier

Bruggeman, Erik

Award date:
1975

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX – NAMUR

Institut d'Informatique

Année académique 1974 – 1975

PROBLEMES DE SECURITE DANS UN ENVIRONNEMENT "TEMPS - REEL"

ETUDE CRITIQUE D'UN

SYSTEME PARTICULIER

Erik BRUGGEMAN

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique

R E M E R C I E M E N T S

C'est très sincèrement que je remercie Monsieur S. de Hepcée, Directeur de ce mémoire, qui m'a fait bénéficier de ses conseils et de son expérience tout au long de cette année.

Je le prie d'être assuré de ma profonde gratitude.

Je tiens à remercier vivement les dirigeants de l'ALLIANCE NATIONALE DES MUTUALITES CHRETIENNES, le groupe UNIVAC de cette société et, tout particulièrement, Monsieur Luyckx pour l'aide qu'il m'a toujours apportée avec la plus grande gentillesse.

Mes sincères remerciements s'adressent également à tous les professeurs de la Faculté d'Informatique de NAMUR, pour la formation et l'esprit d'ouverture sur le monde qu'ils m'ont permis d'acquérir.

Je tiens à remercier, finalement, ceux qui m'ont permis d'assurer la rédaction de ce mémoire, et tout spécialement Madame Jacobs qui a laissé, sur chacune de ces pages, l'empreinte de son travail soigné et efficace.

.....

T A B L E D E S M A T I E R E S

	Page
AVANT-PROPOS.	1
CHAPITRE I. - Le TEMPS-REEL à l'ANMC.	3
Remarque préliminaire	3
Section 1. - Le "software" de base.	4
Section 2. - Le système de contrôle des transactions (RTS)	5
1.2.1. - Système de "paging"	5
1.2.1.1. - notions de page	6
1.2.1.2. - espace virtuel	6
1.2.1.3. - contrôle des pages - les tables	7
1.2.2. - Gestion du CPU.	10
1.2.2.1. - définitions	10
1.2.2.2. - système de contrôle	12
1.2.3. - Cheminement d'une transaction	16
1.2.3.1. - pré-traitement	17
1.2.3.2. - traitement proprement dit	17
1.2.3.3. - post-traitement	20
Section 3. - Description et organisation des fichiers.	23
1.3.1. - Fichiers "système"	23
1.3.1.1. - fichier D Ø	23
1.3.1.2. - fichier F Ø	24
1.3.1.3. - caractéristiques des sous-fichiers "SYSTEME"	25
1.3.2. - Fichiers "application"	25
1.3.3. - Organisation des fichiers "APPLICATION"	26
1.3.3.1. - principe	26
1.3.3.2. - structure du fichier	28

Page

CHAPITRE II. - Les PROBLEMES de SECURITE. 30

Section 1. - Mesures "hardware". 32

2.1.1. - Dédoublement de l'unité centrale 32

2.1.1.1. - configurations à deux CPU's 33

2.1.1.2. - utilisation des périphériques dans un système à deux ordinateurs. 37

2.1.2. - Dédoublement des voies d'accès à un sous-système périphérique. 41

2.1.2.1. - principe du "dual access" 41

2.1.2.2. - "duplex file" 43

2.1.2.3. - opportunité du "dual access" dans les différents sous-systèmes périphériques 45

Section 2. - Mesures "software". 47

2.2.1. - Mesures relatives au traitement interne d'une transaction. 47

2.2.1.1. - la protection mémoire 47

2.2.1.2. - exécution correcte des pages en mémoire 51

2.2.1.3. - les programmes statistiques 56

2.2.1.4. - deux mesures préventives 57

2.2.2. - Mesures relatives à l'exploitation des fichiers. 58

2.2.2.1. - des "handlers" 58

2.2.2.2. - réservation des fichiers "APPLICATION" 61

2.2.2.3. - contrôle des labels 62

2.2.2.4. - règles de modifications d'un fichier "APPLICATION" 64

2.2.3. - Les procédures de secours. 68

2.2.3.1. - types d'erreurs 70

2.2.3.2. - fonctionnement de la "recovery" 71

Page

2.2.3.3. - problèmes de mise au point 75

2.2.4. - Quelques mesures supplémentaires. 82

Section 3. - Mesures "organisationnelles". 84

2.3.1. - Activités concurrentes : le temps-réel et les travaux "batch". 84

2.3.2. - Protection du secret des données. 87

2.3.2.1. - préalables sémantiques 89
2.3.2.2. - schémas de solutions 92

2.3.3. - Mise au point du système. 96

2.3.3.1. - règles de modification 96
2.3.3.2. - inconvénients de la règle 97
2.3.3.3. - considération 97

2.3.4. - Eléments de l'organisation d'exploitation 99

2.3.4.1. - planification des travaux en différé 99
2.3.4.2. - gestion de la bibliothèque 101
2.3.4.3. - programmation de base 102
2.3.3.3. - réflexions 105

CHAPITRE III. - CAS PRATIQUE : SAUVETAGE D'UN FICHIER VOLUMINEUX. 106

Section 1. - Données techniques. 106

3.1.1. - Les bandes 106

3.1.2. - Les disques 108

Section 2. - Procédure par fragmentation du fichier 109

3.2.1. - Estimation du fichier 109

3.2.2. - Procédure 109

Page

3.2.3. - Vidage d'un "positioner module" 111

3.2.3.1. - choix des buffers 111

3.2.3.2. - calcul du temps théorique de vidage d'un PM. 112

3.2.3.3. - taux de transfert des bandes 113

3.2.4. - Organisation du sauvetage du fichier 115

3.2.5. - Administration des "afterlooks" 116

3.2.5.1. - estimation du nombre de bandes nécessaires à l'enregistrement des "afterlooks" d'une journée 116

3.2.5.2. - procédure de gestion des AFL's 118

3.2.6. - Découpe optimale du fichier 121

Section 3. - Les procédures "on-line" 123

3.3.1. - Dump sans réservation du fichier 123

3.3.1.1. - schéma d'exécution 124

3.3.1.2. - remarques 126

3.3.2. - Dump avec réservation du fichier 126

CONCLUSIONS 128

==========

A V A N T - P R O P O S.

Motivations de l'étude.

En ces dernières années, le traitement de l'information par les ordinateurs a subi une évolution profonde par l'introduction des systèmes en "temps-réel". L'apparition de ces systèmes a eu pour effet, notamment, d'accroître l'importance des problèmes de sécurité.

Nous définissons la sécurité comme étant l'ensemble des mesures de protection apportée au système contre les destructions accidentelles ou délibérées d'informations et contre les interruptions dans la continuité des opérations.

Compte tenu de leur importance pratique, il ne semble pas que ces problèmes aient toujours reçu toute l'attention qu'ils méritent. On doit être convaincu que la sécurité concerne un ensemble de mesures que l'on adopte en même temps que l'on décide d'utiliser un ordinateur. Ces mesures font partie intégrante du traitement de l'information.

Objectifs poursuivis.

On peut considérer l'étude comme visant deux objectifs essentiels :

- la compréhension d'un système particulier en "temps-réel"; il s'agira, en l'occurrence, du système en temps-réel de "l'Alliance Nationale des Mutualités Chrétiennes" (ANMC).
- l'analyse des problèmes de sécurité d'un tel système; cette analyse comportera, elle-même, deux objectifs :
  - le premier consistera en une présentation des mesures adoptées par le Projet à l'ANMC.

- le second soulignera les difficultés concernant les problèmes évoqués. Certaines propositions seront suggérées vis-à-vis de problèmes non encore résolus ou incomplètement élaborés.

Cette étude peut paraître ambitieuse; en fait, tous les aspects de la sécurité ne sont pas abordés, leur domaine étant trop vaste. Deux orientations ont été suivies dans l'analyse. Elles consistent à considérer le problème :

- du point de vue "service"

  cet aspect concerne les mesures apportées afin de rendre, à l'utilisateur, un service ininterrompu.

- du point de vue des fichiers

  cet autre aspect vise l'intégrité des informations dans les fichiers.

Afin d'ajouter, à cette étude, un caractère "expérimental", une recherche plus approfondie a été entreprise afin de résoudre un problème crucial se posant au Projet.

Il s'agit du sauvetage d'un fichier volumineux.

==========

C H A P I T R E I.

LE TEMPS-REEL A L'A.N.M.C.

Remarque préliminaire.

L'objectif final du Projet est un système multi-processeur. Il s'agira d'une configuration de trois ordinateurs dont les fonctions respectives seraient :

- la gestion des communications,
- le traitement du temps-réel,
- les travaux différés,

en précisant qu'un des CPU's, différent de celui du traitement du temps-réel, servirait de "back-up" de ce dernier.

Plusieurs étapes sont nécessaires à cette réalisation. Précisons que le travail se basera essentiellement sur l'étape où n'apparaissent encore que 2 CPU's. Le système décrit un traitement multifils, par opposition à un traitement monofil. Dans ce dernier cas, le travail des programmes d'application sur un message est exécuté complètement avant que le message suivant ne soit pris en charge; les messages sont traités en série.

En traitement multifils, différents programmes travaillent parallèlement sur différents messages, le contrôle passant de l'un à l'autre suivant certaines conditions déterminées par les événements extérieurs.

Notons, par ailleurs, que le Système est prévu pour supporter un réseau volumineux de terminaux ($\pm$ 300) répartis dans toute la Belgique. L'application elle-même a pour but essentiel, la consultation et la mise à jour des fichiers "application".

Pour tous les termes techniques utilisés dans l'analyse, on peut se référer au glossaire se trouvant en fin de dossier.

Section 1. - Le "software" de base.

Le "software" standard fourni par le constructeur est le "Real Time Operating System" (RTOS).

Comme la plupart des "software" standards, le RTOS s'est avéré trop volumineux et très lourd à l'exécution. Etant donné également la charge prévue pour le système, il a été nécessaire de modifier le RTOS. C'est ainsi que certaines routines n'ont pas été générées; certaines ont été remplacées par d'autres, plus performantes; d'autres, enfin, ont fait l'objet de modifications uniquement. Les routines ainsi retravaillées ont été regroupées sous un seul run appelé "Real Time System" (RTS). Les routines RTS concernent principalement le système de contrôle des transactions et la gestion de la mémoire virtuelle ("paging").

Pourquoi le RTS est-il un run ?

En fait, toute autre forme eut été possible, notamment une extension du RTOS. Le seul avantage est qu'un run peut être chargé à n'importe quel endroit en mémoire centrale, de sorte qu'au chargement, il ne nécessite pas l'arrêt des autres runs ("batch") présents en mémoire centrale à ce moment-là.

Le fait qu'il puisse être chargé et déchargé implique également une meilleure exploitation de la place "mémoire".

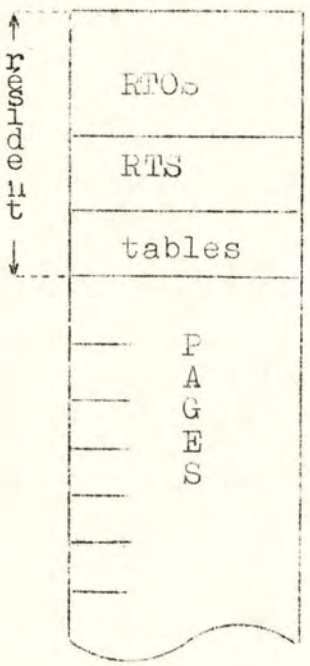
Section 2. - Système de contrôle des transactions (RTS).

En abordant cette section, notre intention sera, avant tout, de donner au lecteur une compréhension suffisante quant aux mécanismes de traitement du temps-réel. La description qui va suivre ne se veut pas complète mais néanmoins suffisante pour le reste de l'analyse.

1.2.1.- Système de "paging".

La mise au point d'un tel système s'est révélée nécessaire afin d'optimiser les performances.

Le mérite en revient aux membres du Projet qui ont entièrement conçu ce système de mémoire virtuelle.



- Le RTOS, certaines routines du RTS et les tables du système forment la partie résidente de la mémoire.
- Le reste de la mémoire constitue la partie dynamique. Elle est allouée aux pages.
- UNIVAC n'utilise pas le principe de partition de la mémoire. C'est ainsi que les travaux "batch" se déroulent dans la partie de mémoire laissée libre par les pages.
- La mémoire principale 96-K (mot) se décompose en bancs de 32 K accessibles de façon indépendante les uns des autres. Chaque banc est constitué de 8 segments (ou "bays") de 4-K. Une "bay" contient 4 pages de 1-K (mot).

1.2.1.1. - Notions de pages.

page "hardware"

un segment de mémoire centrale de 1-K dont l'adresse absolue de début est un multiple de 02000 (octal). Cette zone mémoire peut contenir une page "software", une page "data", une page de blocs de contrôle ("packets") ou une TAR (espace de travail d'une transaction).

page "virtuelle"

une page d'instructions localisée sur tambour.

page "software"

une page "hardware" contenant une page "virtuelle" ou une page "virtuelle" chargée en mémoire.

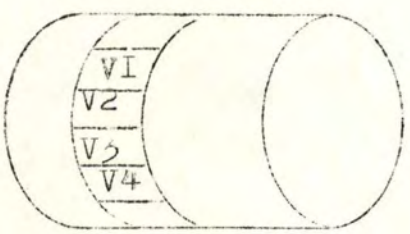
page "data"

une page "hardware" contenant des données. Il s'agira :
- soit d'une FAR (données d'un fichier "application")
- soit d'une page contenant des données spécifiques quelconques.

TAR

une page "hardware" établie pour le traitement d'un message ou d'un enregistrement.

1.2.1.2. - Espace "virtuel".



- tambour -

. Il y a, sur tambour, quatre versions d'une même page.

. V_1, V_2, V_3, V_4 sont quatre pages virtuelles identiques.

. Si l'on désire charger une page virtuelle dans une page "hardware" dont l'adresse, à l'intérieur d'un segment, est "02000", la deuxième version sur tambour de cette page virtuelle sera choisie pour effectuer le chargement. On réduit, de ce fait, l'effort de relogement. En effet, il suffira d'ajouter, à chaque adresse, la constante de relogement constituée du numéro de la "bay" dans laquelle a été chargée la page virtuelle.

1.2.1.3. - Contrôle des pages. Les tables.

1. VTAB (virtual table)

Cette table se compose d'un mot par page virtuelle.
Ce mot contient :

- des informations concernant l'état de cette page;
- un pointeur.

Le pointeur dépend de l'état de la page :

- si la page est en mémoire centrale ou occupée à être chargée en mémoire, le pointeur consistera en l'adresse de l'entrée, de la page "hardware" correspondante dans HTAB.
- si la page n'est pas en mémoire, le pointeur sera constitué de :
  . zéro si aucun appel, pour cette page, n'a été enregistré.
  . l'adresse vers une entrée dans PHTAB lorsque un ou plusieurs appels ont été enregistrés (cas où il n'y a plus de page "hardware" libre).

2. HATB

Une table des pages "hardware" se compose d'entrées de 8 mots.

Ces 8 mots contiennent notamment :
- l'adresse de l'entrée de la page dans VTAB (Ø77ØØØØ dans le cas où la page "hardware" est libre),
- des pointeurs vers le premier et le dernier bloc de contrôle pour cette page.

On définit un bloc de contrôle ("activity packet") comme étant tout appel d'une transaction pour cette page.

Physiquement, il s'agit d'un enregistrement de 32 mots comprenant l'adresse virtuelle de la page, les limites de la TAR qui demande l'exécution de cette page et le contenu des registres du système.

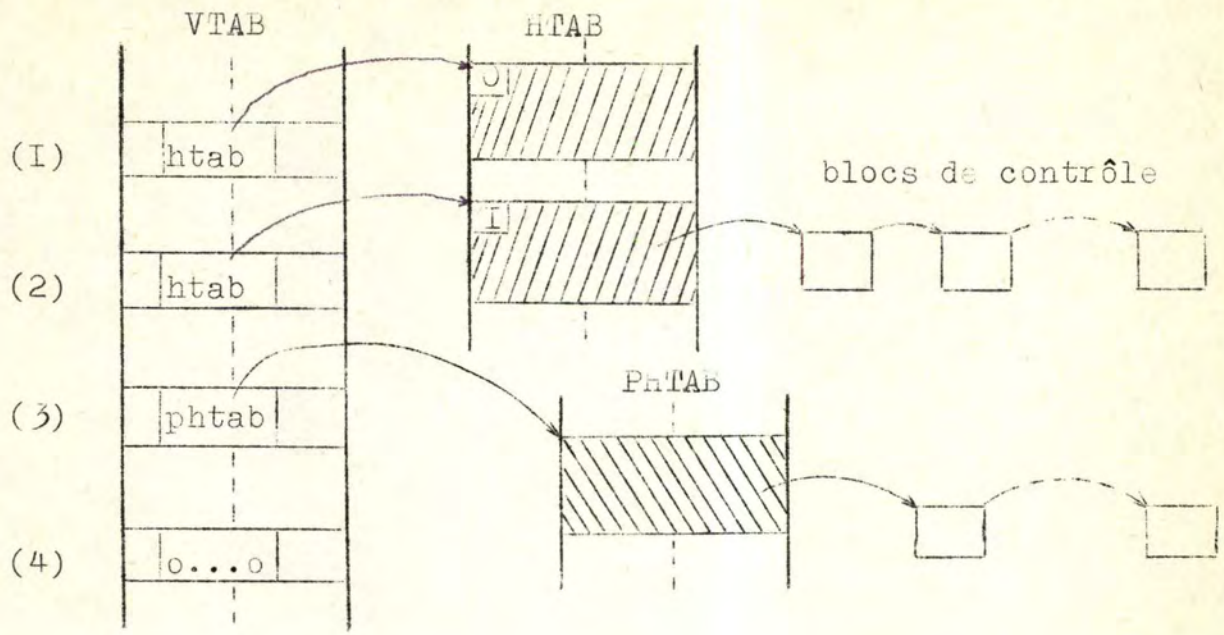
3. PHTAB.

(pseudo hard table)

Cette table a le même format que la HTAB. L'objectif de cette table est de prendre en considération les appels concernant des pages virtuelles qui ne sont pas disponibles en mémoire centrale.

C'est le cas où plus aucune page "hardware" n'est libre.

4. RELATIONS ENTRE LES TABLES.



Ier cas : la page est utilisable et déjà en mémoire. Les blocs de contrôle sont directement chaînés à la file d'attente du distributeur de tâches (SWITCHER)

2me cas : la page est utilisable, mais le chargement dans la page "hardware" n'est pas terminé. Les blocs de contrôle sont chaînés à l'entrée correspondante dans HTAB.

3me cas : la page n'est pas disponible (cas où il n'y a plus de page "hardware" libre). Les appels pour cette page virtuelle sont chaînés à l'entrée correspondante dans PHTAB.

4me cas : si le pointeur dans VTAB est zéro, aucun appel pour cette page n'a été enregistré.

(Le fonctionnement du SWITCHER sera décrit plus en détail ultérieurement.)

1.2.2. - Gestion du CPU.

Définissons, d'abord, ce qu'il faut entendre par fonction et activité.

Ces notions sont fondamentales pour la suite de l'exposé.

1.2.2.1. - Définitions.

A) Fonction.

Suite d'instructions dont le résultat est un changement :
- de certaines données;
- de l'état du système.

Types de fonctions.

1. Fonctions "SYSTEME" : ensemble de facilités mis à la disposition de l'utilisateur.
   (exemples) - allocateur de pages
   - traitement des adresses "retour"
   - routines de traitement des interruptions

   Ces fonctions sont généralement résidentes.

2. Fonctions "PROCESSEUR" : ensemble de routines qui exécutent le cheminement de la transaction à travers le système.
   (exemples) - routine de gestion du réseau de terminaux.
   - routine de gestion des files d'attente.
   - routine d'initialisation du traitement d'un message ("transaction starter")

3. "TPS" ("transaction processing segment") : il s'agit des programmes exécutant le traitement proprement dit des messages.

Remarque.

La fonction sera constituée d'une page ou d'un groupe de pages.

B) Activité.

Il s'agit d'une demande pour une fonction.

Types d'activités.

1. Activités "SYSTEME" : demandes d'exécution pour les fonctions "SYSTEME". Ces demandes sont représentées par des blocs de contrôle ("packets"). Ces activités sont gérées par le séquenceur du RTS.
2. Activités "PROCESSEUR" et "TPS" : demandes d'exécution d'une fonction "PROCESSEUR" ou d'un "TPS". Les demandes sont également représentées par des blocs de contrôle mais elles sont gérées par le SWITCHER du RTS.

Etats possibles d'une activité.

Une activité peut être :

1. en contrôle (ACTIVE STATE), c'est-à-dire en exécution;
2. en attente de contrôle dans le "SWITCHER" ou le séquenceur (READY STATE);
3. en attente d'un appel pour cette fonction (WAIT STATE).

Lorsqu'une activité n'est pas dans l'état actif, elle se trouve dans une file d'attente. Cette file d'attente peut être celle . du SWITCHER
. du séquenceur
ou toute autre file d'attente, telle que :

- la file d'attente des demandes pour une page "data"
- la file d'attente des appels pour une page virtuelle
- la file d'attente de traitement des I/O's.

Remarque.

La notion d'activité porte à confusion. La définition qui vient d'en être donnée se raccroche à celle de "bloc de contrôle" vue précédemment. Par contre, si

l'on se place au niveau du RTOS, l'activité se définira comme une séquence d'instructions logiquement ininterruptible (tâche).

1.2.2.2. - Système de contrôle.

1. Les activités "SYSTEME" sont gérées par le séquenceur; celui-ci opère en conjonction avec une table de modules. Chaque module représente une fonction "SYSTEME" différente.
   Les différents appels (les activités) pour un même module sont chaînés entre eux.

2. Les activités "PROCESSEUR" et TPS sont gérées par le "SWITCHER". Celui-ci possède trois files d'attente avec des priorités distinctes.
   Ces files d'attente correspondent à trois types d'activités :
   a) les activités "processeur" ..... priorité 1.
   b) les TPS's ..... priorité 2.
   c) les DBJ's (data base jobs, travaux "batch" dont on verra l'utilité ultérieurement)..... priorité 3.

   Le "SWITCHER" effectue sa gestion de la manière suivante :
   - il analyse la file la plus prioritaire;
   - dans une même file d'attente, la règle de sélection utilisée est FIFO (first in, first out).
     S'il n'existe aucun appel de fonction dans cette file, l'analyse se poursuit dans la file de priorité directement inférieure.

3. Fonctionnement.

- Comme déjà dit précédemment, les blocs de contrôle, c'est-à-dire les appels de fonctions, sont chaînés à la HTAB aussi longtemps que la page virtuelle en question n'a pas fini son chargement en mémoire. Or, dès que

celui-ci se termine, la chaîne des appels pour cette page se voit automatiquement reliée au "SWITCHER" dans une des trois files de ce dernier. Ces activités, en effet, sont maintenant prêtes à l'exécution.

- Si une certaine activité a le contrôle et que, durant son exécution, apparaît une opération d'entrée/sortie, le contrôle sera rendu au "SWITCHER". Celui-ci fera une nouvelle sélection dans ses files d'attente et donnera le contrôle à une autre activité dont la page "software" est peut-être identique à celle qui était exécutée précédemment (notion de Ré-entrance).
  La plupart des fonctions "processeur" et les TPS sont ré-entrantes.

- Si, dans l'exécution d'une page, on aboutit à une condition d'erreur, un bloc de contrôle sera ajouté à la file d'attente du module du séquenceur traitant le type d'erreur survenue.
  De même, lorsque survient une fin d'opération d'entrée/sortie, une demande supplémentaire sera introduite dans la file d'attente du module séquenceur exécutant les terminaisons d'E/S.
  Il en ira de même pour tout appel "superviseur" rencontré dans l'exécution d'une page.

- Lorsque le séquenceur reçoit le contrôle (soit par le TIMER, soit après que le RTOS ait reçu le contrôle), il en profite pour exécuter toutes les demandes qui sont chaînées à ses différents modules.
  Lorsque son travail est terminé, il rend le contrôle au "SWITCHER" afin que celui-ci détermine quelle sera la page suivante à exécuter.

4. Relations entre le RTS et le RTOS.

Remarques.

a) Il a été question, jusqu'à présent, du fonctionnement à l'intérieur du RTS. On a défini, pour cela, deux distributeurs de travaux : le séquenceur et le "SWITCHER". Le RTOS, lui, possède également un distributeur de travaux que l'on appelle aussi "SWITCHER"; d'où la confusion qui peut apparaître par le double emploi de ce mot.

b). Chaque activité se voit attribuer une priorité. Cette priorité est déterminée par la classe et le "mid".

- chaque run (un ou plusieurs programmes) possède une classe.
- chaque programme (une ou plusieurs activités) possède un "mid" (mode indicator).

. Classe et "mid" sont combinés pour former une priorité d'activité suivant la relation :

$$\text{priorité} = \text{classe} + \text{nombre de classes} * (\text{mid} - 1)$$

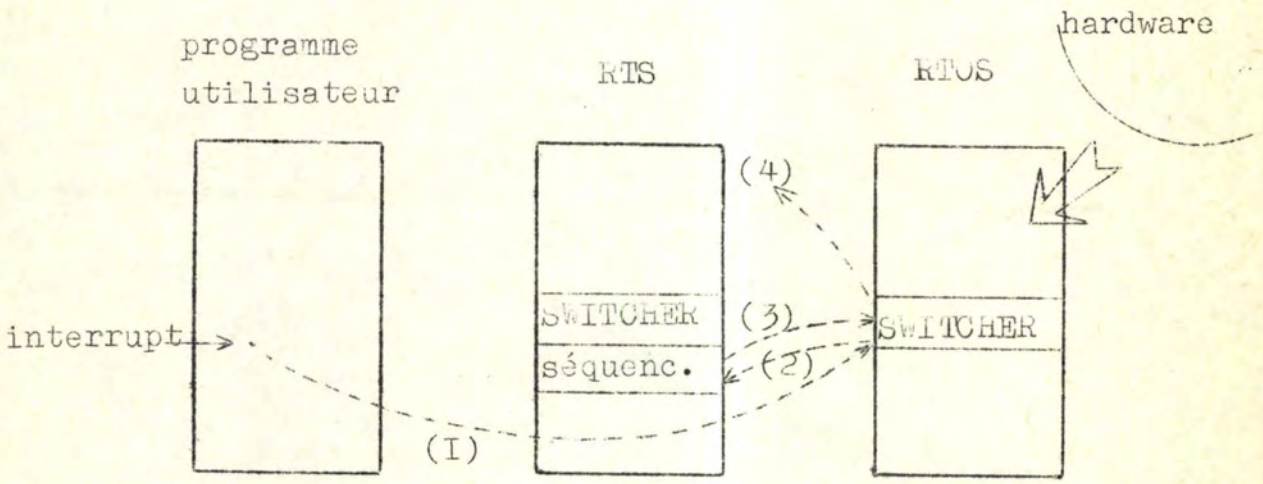
. Le run RTS, lors de son chargement, est de priorité "1" (classe 1 et mid 1), alors que le RTOS s'exécute à la priorité "0". Mais après son initialisation, le RTS sera considéré comme faisant partie du RTOS et on lui donne la priorité "0" également (priorité la plus élevée).

. La classe "3" est réservée aux travaux en différé, tandis que la classe "2" n'est pas utilisée.

Les relations entre le RTS et le RTOS se font par le mécanisme des interruptions. On distinguera deux types d'interruption :

- du type "hardware" (ex. une fin d'I/O)
- de type "appel superviseur".

Seul, le premier type d'interruption fera intervenir le RTOS.



- Soit qu'une interruption intervienne au moment de l'exécution d'un programme (une page du RTS ou un programme "batch").

- L'interruption de type "hardware" est détectée par le RTOS (1).

- Celui-ci sauve l'environnement du programme interrompu car chaque activité d'un programme utilise le même jeu de registres.

- Le RTOS analyse l'interruption; s'il découvre que celle-ci (une fin d'I/O par exemple) concerne le temps-réel, il enverra un bloc de contrôle dans le module séquenceur du RTS traitant les fins d'I/O (2).

- Le "SWITCHER" du RTOS effectue alors une analyse (3) dans ses files d'attente afin de choisir quelle sera la prochaine activité qui recevra le contrôle (d'après les priorités).

  Si le programme qui a été interrompu est un programme "batch" (priorité la plus basse), il y a peu de chance que le contrôle lui revienne après le traitement de l'interruption.

1.2.3. - Cheminement d'une transaction.

Il s'agit, ici, de décrire les différentes phases de traitement d'un message, depuis la prise en charge jusqu'à l'émission du message de sortie vers le ou les terminaux de réception.

Ces différentes phases sont exécutées par les routines du RTS. Ces routines sont, elles-mêmes, soit résidentes, soit sous la forme de pages.

Le trajet d'un message au travers du système se décompose en trois étapes (fig. 1.2.3.a.) :

- pré-traitement
  - prise en charge du message
  - mise en file d'attente des messages "input"
  - sélection d'un message dans la file d'attente

- traitement
  - exécution de la transaction du niveau de l'application.

    L'outil de cette exécution est le TPS (transaction processing segment).

- post-traitement
  - mise en file d'attente du message "output"
  - émission du message "output" vers le ou les terminaux de réception.

1.2.3.1. - Réception du message et mise en file d'attente. PRE-TRAITEMENT.

. La prise en charge est réalisée par le MCH ("Message Communication Handler"). Le MCH est résident en mémoire. Il effectue une demande d'un buffer de 1-K à l'allocateur de pages "hardware".

. Si la demande est satisfaite par l'allocateur de pages, le MCH place le message "input" dans ce buffer. Dans le cas contraire, la demande est placée dans la file d'attente de l'allocateur de pages.

. Le MCH fait appel, ensuite, au MQH ("Message Queing Handler") dont le but est d'écrire le contenu du buffer à un endroit défini (FH-TAR) sur tambour. A chaque terminal est associée une zone-tambour FH-TAR.

1.2.3.2. - TRAITEMENT.

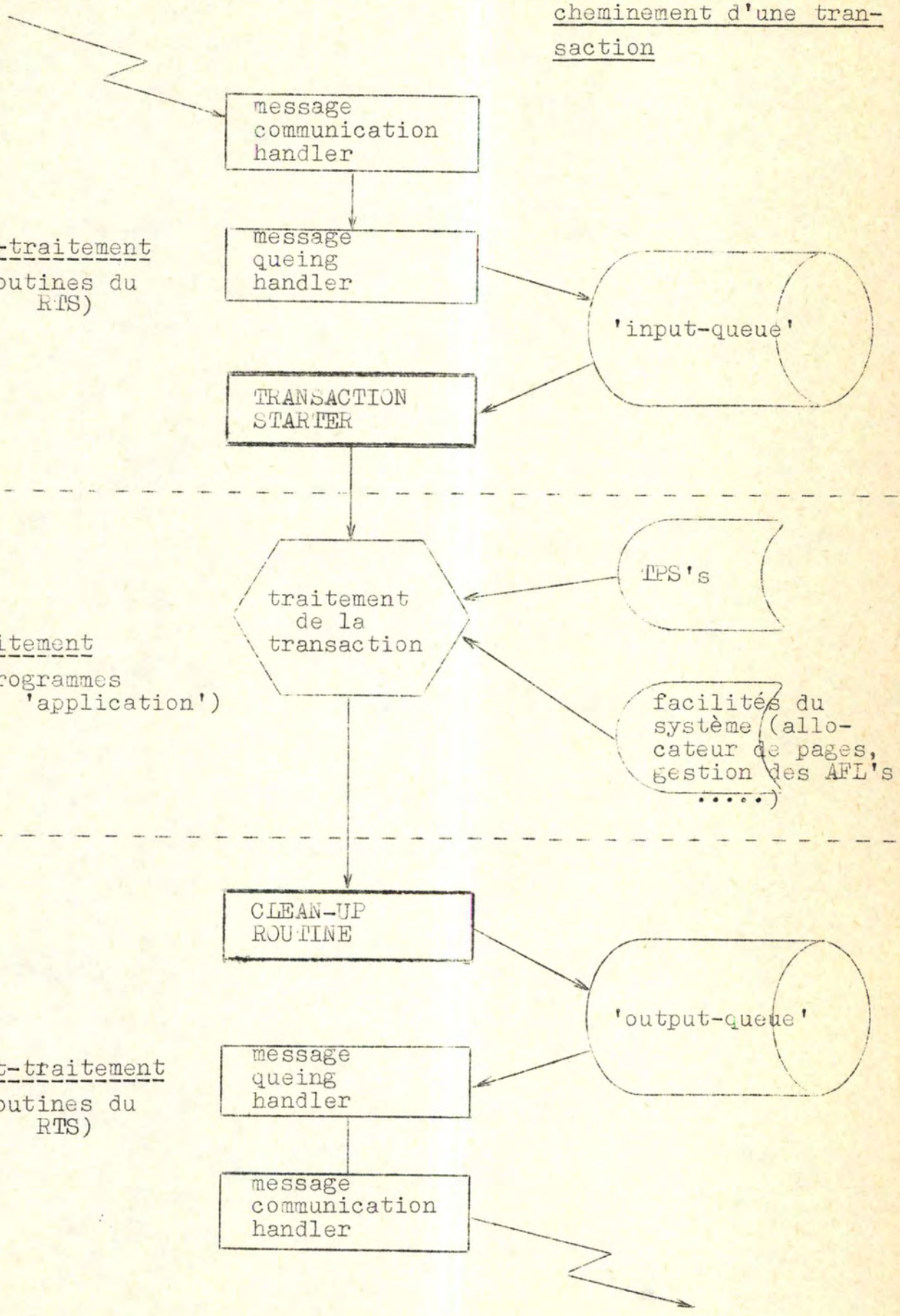
A. Une routine spéciale (TRANSACTION STARTER) analyse à intervalles réguliers le contenu de la file d'attente des messages "input".

. S'il existe un message dans l' "input queue", il exécute les fonctions suivantes :

- réservation d'une zone 1-K en mémoire centrale (TAR)
- transfert du contenu de la FH-TAR correspondante dans la TAR.

fig.I.2.3.a

cheminement d'une transaction



- première analyse du message (S'agit-il d'un message de service ou d'une transaction ? Le message répond-il aux exigences de format ? Le format se trouve-t-il dans la partie M?WORK (zone de travail) de la TAR, ...)

B. Le traitement proprement dit est exécuté par le programme utilisateur.

- On appelle "transaction monitor" l'ensemble des TPS's nécessaires au traitement du message. Plusieurs TPS's (programmes d'application) sont peut-être nécessaires. Tout programme utilitaire se trouve sur tambour sous forme de pages "application"..

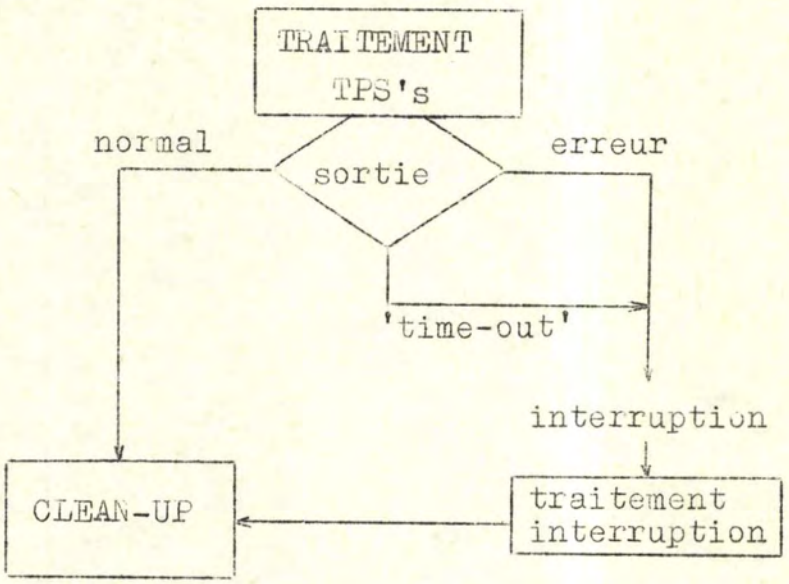
- Les TPS's ont, comme objectifs, la consultation et la mise à jour des fichiers applications. Ils effectuent ces activités par l'intermédiaire des "file Handlers". Ces "handlers" constituent des fonctions systèmes.

C. La routine "CLEAN-UP" clôture le traitement de la transaction. Elle effectue les fonctions suivantes :

- commander l'archivage sur bande "LOGGING" d'un certain nombre d'informations concernant la transaction qui se termine;
- désallocation de la place mémoire occupée par la transaction (TAR, page-data, FAR, ...)
- libération des éléments des fichiers qui étaient concernés par la transaction;
- activation du MQH ("Message Queing Handler") pour écrire la TAR (qui contient le message de sortie) sur tambour.

Remarque.

En fait, le traitement proprement dit peut se terminer de trois manières différentes :



La sortie peut se faire :

- normalement
- par un "TIME-OUT" (ex. transaction qui boucle)
- par une erreur

Dans les deux derniers cas, une routine spéciale de traitement de l'interruption sera exécutée avant de passer, comme dans le premier cas, à la routine de "clean-up".

1.2.3.3. - POST-TRAITEMENT.

. Le scénario est inverse de la première phase.

. Le MQH, averti par la routine de "CLEAN-UP" , exécute le transfert de la TAR dans la ou les FH-TAR's, suivant qu'il y a un ou plusieurs terminaux de

réception. La ou les FH-TAR's sont placées dans la file d'attente des messages OUTPUT.

. Il appartient au MCH d'examiner à intervalles réguliers la file d'attente des messages de sortie et d'exécuter la sortie du message vers le terminal de réception.

Pour ce faire, il effectue :
- le transfert de la FH-TAR vers la TAR (en mémoire centrale);
- l'envoi vers le terminal de réception.

Notes.

1.- MCH, MQH, TRANSACTION STARTER, CLEAN-UP constituent des fonctions "PROCESSEUR".

- Les facilités offertes par le RTS, telles que les "handlers" et les routines de traitement des interruptions, constituent des fonctions "SYSTEME".

- Les TPS's qui effectuent le traitement proprement dit, sont formés par des pages applications.

2. Une partie importante du RTS est la "recovery". Celle-ci est basée sur l'écriture (sur bande magnétique) des enregistrements du fichier qui ont fait l'objet d'une mise à jour (AFTERLOOKS).

La procédure du "recovery" est une combinaison de rechargement des fichiers à un certain point et de mise à jour de ceux-ci par tous les changements encourus après ce moment.

La "recovery" des fichiers est aussi concernée lorsque des parties de fichiers deviennent illisibles.

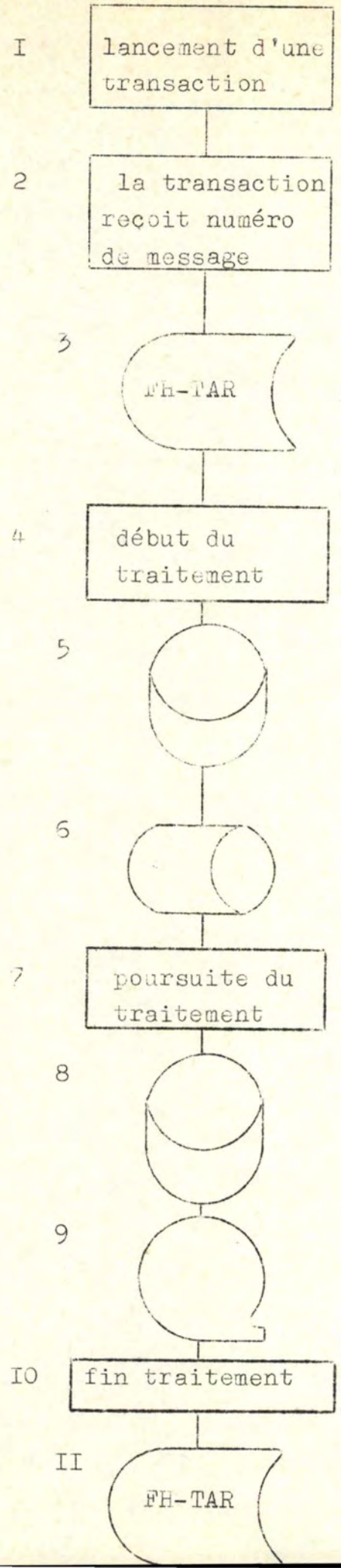


fig.I.2.3.b

étapes d'une mise à jour d'un enregistrement

la transaction et son numéro de message sont placés en 'input-queue'

lecture de l'enregistrement a mettre à jour

écriture de l'enregistrement et du numéro de message sur le fichier 'beforelook'

mise à jour de l'enregistrement

écriture de l'enregistrement mis à jour et du numéro de message sur le fichier 'afterlook'

la transaction est placée en 'output-queue'

Dans certains cas, la "recovery" sera initialisée par la bande "LOGGING".

La "recovery" des transactions est basée sur les BEFORELOOKS, conservés sur tambour. Ceux-ci permettent de redémarrer les transactions qui étaient actives au moment de l'interruption.

L'utilisation des enregistrements "beforelooks" et "afterlooks" est décrite par la fig. 1.2.3.b.

Section 3. - Description et organisation des fichiers.

Remarque préliminaire.

Il sera question, tout au long de cette analyse, de fichiers systèmes et de fichiers applications.

Les notions de fichiers "data" et de fichiers du temps-réel seront évitées car elles portent à confusion.

On définit fichier "système" et fichier "application" selon qu'ils contiennent des informations (données ou programmes) appartenant soit au système, soit aux utilisateurs.

1.3.1. - Fichiers "SYSTEME".

1.3.1.1. - Fichier D Ø

Ce fichier, conservé sur tambour, se compose de plusieurs sous-fichiers :

1. TPS : ce fichier contient tous les programmes de traitement des transactions.

2. RCV : fichier "recovery" contenant toutes les données critiques indispensables en cas d'interruption du système.

3. DRMLOG : fichier des "afterlooks" et des enregistrements du "logging".

4. SCRATCH : zone fixe sur tambour, allouable aux utilisateurs et permettant à ces derniers de conserver des résultats intermédiaires.

5. FH-TAR : fichier comprenant, pour chaque terminal, une zone fixe de 1-K mots.

6. FH-TARJ : fichier des zones "tambour" allouées aux travaux DBJ's (Data-Base Jobo : programmes "batch" exécutés sous le RTS et utilisant les ressources de celui-ci).

7. BFLOOK : fichier des "beforelooks".

1.3.1.2. - Fichier F Ø

F Ø est implémenté sur disque (auparavant sur tambour "Fastrand") et se décompose en 4 sous-fichiers :

1. JRPTY : ce fichier contient toutes les informations concernant les programmes "batch" se déroulant durant le temps-réel.

2. TLOG : ensemble d'informations concernant les bandes "logging" non clôturées. En cas d'interruption, on n'a pas le temps de clôturer les bandes "logging". TLOG contient les informations permettant de retenir l'environnement de la situation interrompue.

3. STAT : fichier contenant toutes les données statistiques.

4. ABORTT : ce fichier prend en considération tous les numéros de messages concernant les transactions interrompues.

1.3.1.3. - Caractéristiques des sous-fichiers "SYSTEME".

fichier	sous-fichier	état	copie
D Ø	TPS	ROF	S
	RCV	RWF	d
	DRMLOG	RWF	d
	BFLOOK	RWF	d
	SCRATCH	RWF	d
	FH-TAR	RWF	d
	FH-TARJ	RWF	d
F Ø	JRPTY	RWF	d
	TLOG	RWF	d
	STAT	RWF	d
	ABORTT	RWF	d

S = Simple
d = double

ROF = fichier de lecture

RWF = fichier de lecture et d'écriture

Note.
Les fichiers, conservés en deux copies, sont appelés "fichiers critiques".

1.3.2. - Fichiers "APPLICATION".

Il existe deux grands fichiers "APPLICATION", eux-mêmes décomposables en une série de sous-fichiers distincts.

Il n'est pas indispensable, dans le cadre de cette analyse, de décrire la nature des différents enregistrements. Signalons cependant l'importance de la notion de "module".

Exemple.

L'enregistrement d'un individu contiendra les modules suivants :

- nom
- prénoms
- adresse
- état civil
- organisme assureur

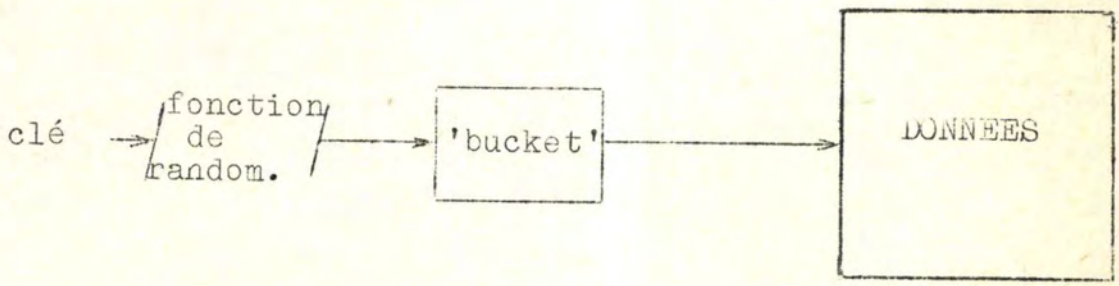
On peut définir un module comme une unité indécomposable d'information. Il y a une structure de ces modules. Certains de ceux-ci ne sont accessibles que si d'autres l'ont été précédemment.

1.3.3. - Organisation des fichiers "APPLICATION".

Tous les fichiers "APPLICATION" sont organisés selon les mêmes principes.

L'organisation utilisée est une organisation en "tiroirs". L'accès à un tiroir est obtenu par une fonction de randomisation. Les tiroirs ("buckets") réalisent une partition du fichier.

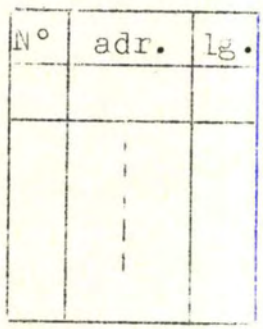
1.3.3.1. - Principe.



- Chaque enregistrement est défini de façon unique par une clé qui est le numéro national d'un individu.
  Pour accéder à un enregistrement, cette clé est divisée par le nombre de "buckets". Le reste de cette division détermine le "bucket" qu'il faudra analyser afin de trouver l'adresse de l'enregistrement dans la zone "données" du fichier.

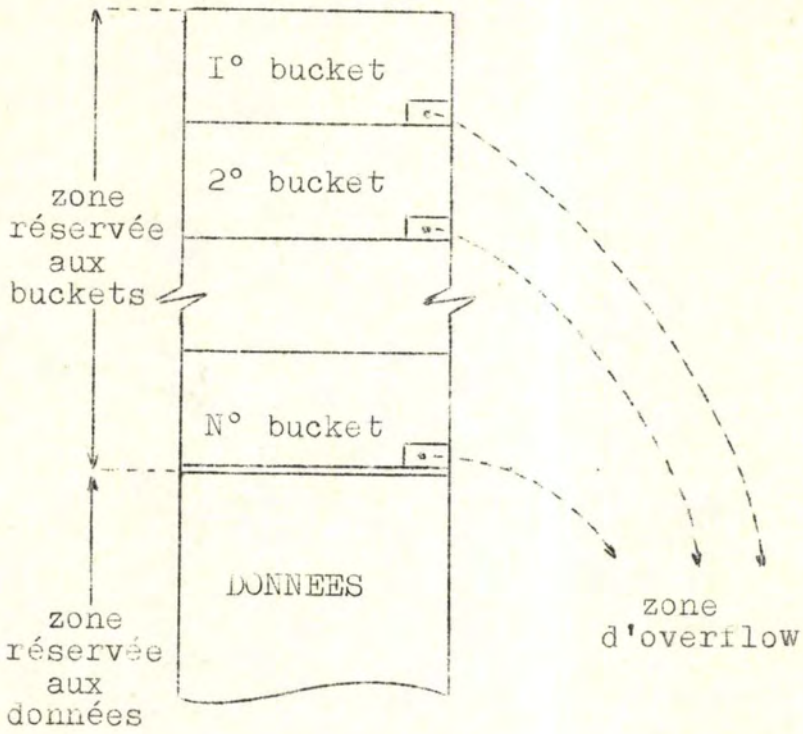
- Cette organisation en "tiroirs" permet d'accéder plus rapidement aux enregistrements. En effet, la transformation de la clé précise immédiatement le "bucket" à considérer. Il suffira d'effectuer la recherche dans un seul "bucket".

- Le "bucket" contient trois données :
  - la clé (numéro national).
  - l'adresse de l'enregistrement sur disque.
  - la longueur de l'enregistrement.



(description d'un "bucket")

1.3.3.2. - Structure du fichier.



- Chaque "bucket" comporte 55 entrées et un pointeur vers une zone d'overflow. En effet, la transformation réalisée sur la clé n'interdit pas un nombre d'entrées supérieur à 55 dans un même "bucket".

- Organisation de la zone "données" : cette zone est également subdivisée en parties.
  En effet, au fil de l'exploitation, des enregistrements se modifient; d'autres s'ajoutent; certains sont supprimés.

  Le critère de décomposition de la zone "données" est un critère de longueur des enregistrements. Un enregistrement, dont la longueur augmente dans une certaine norme pré-établie, se verra transféré dans une autre

zone destinée à recevoir des enregistrements de cette longueur, l'ancienne place occupée par l'enregistrement devenant libre.

Cette organisation exige :

1. une tenue à jour des emplacements libres du fichier;
2. une mise à jour des adresses dans les "buckets";
3. des réorganisations régulières (suppression d'informations devenues inutiles).

==========

C H A P I T R E II.

LES PROBLEMES DE SECURITE.

Les problèmes de sécurité, dans le cadre de cette étude, seront traités sous deux angles différents :

- la protection des fichiers
- et la continuité du service.

La protection des fichiers, c'est-à-dire leur conservation, englobe les mesures visant à assurer l'intégrité des informations. Il s'agit de considérer l'ensemble des procédures de contrôle garantissant la validité des données.

La continuité, dans le service, concerne les mesures permettant une reprise normale des opérations en cas d'incident. Le problème de la continuité, dans le service, implique un choix. Une première solution consiste à continuer l'exploitation tant qu'il est possible de le faire, quel que soit le type d'erreur ou de panne survenu, et avec les risques (perte d'information) que cela peut comporter. Une seconde solution préconise un arrêt complet de l'exploitation jusqu'à la réparation de la panne.

Pour illustrer ces deux façons de procéder, considérons le cas d'une panne survenant sur une copie d'un fichier tenu en deux exemplaires :

- si l'on décide de continuer l'exploitation, celle-ci reposera sur l'emploi d'une copie unique; la reconstitution de la seconde sera effectuée ultérieurement. Les risques sont, dès lors, considérables. En effet, si la copie restante subit un sort identique à la première, les pertes en information seront immenses.
- si, par contre, les opérations sont suspendues, la copie défectueuse sera immédiatement reconstituée à partir de

la copie intacte. Il n'y a aucun risque de perte d'information mais seulement une influence défavorable sur le temps de réponse.

Dans le contexte particulier du Projet, la seconde solution sera préférée. En effet, si le temps de réponse est important, il n'est cependant pas critique au point d'interdire une suspension temporaire des opérations dans la mesure où celle-ci reste dans des limites raisonnables (quelques minutes).

Notons que ce choix ne sera pas acceptable dans n'importe quel système en temps-réel. Certains systèmes exigent un temps de réponse très rapide au-delà duquel l'information perdrait toute valeur.

.....

Section 1. - Mesures "hardware".

Les mesures, purement "hardware", prises par le Projet dans le cadre d'une meilleure sécurité, sont relatives au dédoublement de certaines unités telles que l'unité centrale et les unités d'accès aux périphériques.

Spécifions, avant, les quatre fonctions que le système se doit de remplir indépendamment de sa configuration :

Fonctions du système.

1. le contrôle des communications (C)
2. le traitement interne des transactions (RT)
3. le traitement des "Data Base Jobs" (DBJ) : il s'agit de programmes "batch" exécutés sous le contrôle du RTS et qui profitent des facilités offertes par ce dernier.
4. le traitement des travaux en "batch pur" (B). Les programmes ne sont pas sous le contrôle du RTS; ils ne se déroulent pas sous forme de pages. Ces travaux n'ont, en principe, pas le droit de modifier les fichiers "application". Nous verrons, par la suite, qu'ils constituent un problème crucial pour le système.

2.1.1. - Dédoublement de l'unité centrale.

Dans le cadre d'un système comprenant deux unités centrales, sept configurations différentes sont susceptibles d'être réalisées.

Le tableau, ci-après, reprend les différentes configurations qui se différencient par le fait que les quatre fonctions du système, énoncées précédemment, y sont réparties de façons différentes :

	B	DBJ	RT	C	(soient CPU.A et CPU.B)
1	A	A	A	B	FRONT END
2	A	A	B	A	RT - B/DBJ/C
3	A	B	A	A	DBJ - RT/B/C
4	B	A	A	A	B - RT/DBJ/C
5	A	A	B	B	RT/C - DBJ/B
6	A	B	A	B	RT/B - C/DBJ
7	B	A	A	B	B/C - RT/DBJ

Notes.

1) Le "software" utilisé permettra toutes les configurations. Le passage d'une configuration à une autre ne posera aucun problème particulier.

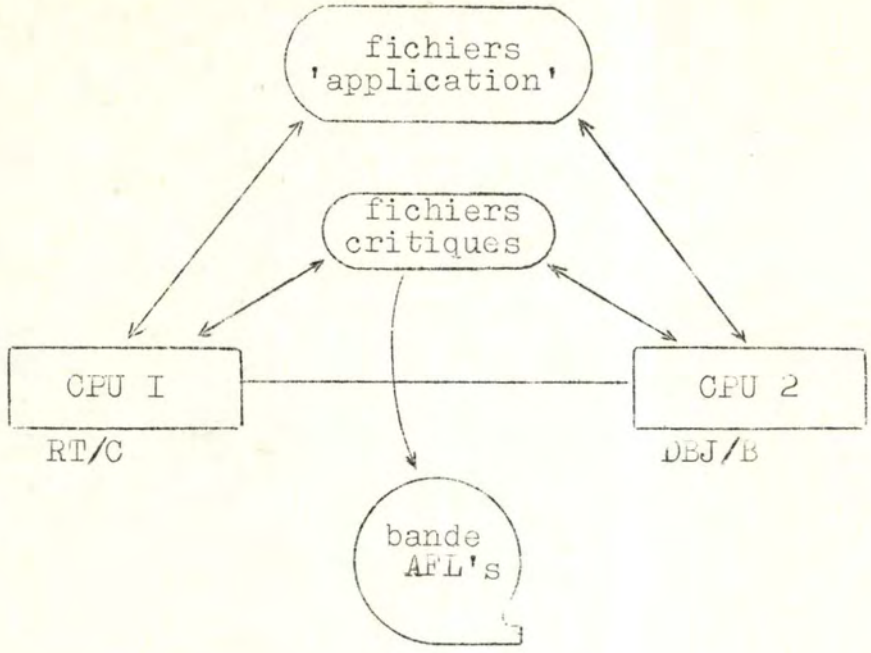
2) L'objectif premier est de disposer d'un CPU supplémentaire, capable de reprendre l'exploitation du "temps-réel" en cas de défaillance de l'unité réservée à cette tâche.

2.1.1.1. - Configuration à deux CPU's.

Deux configurations sont généralement retenues : il s'agit des configurations 5 et 7.

Analysons plus en détail ces deux types de configuration :

A) Première configuration : RT/C - DBJ/B



- soient le CPU.1 traitant le temps-réel et gérant le réseau de communications,
  le CPU.2 traitant les DBJ's et le "batch pur".

- les deux unités centrales ont la possibilité de faire des consultations et des mises à jour dans les fichiers "application". De même, chacun d'eux aura à utiliser les fichiers critiques du système.

- Une telle configuration pose quelques problèmes :

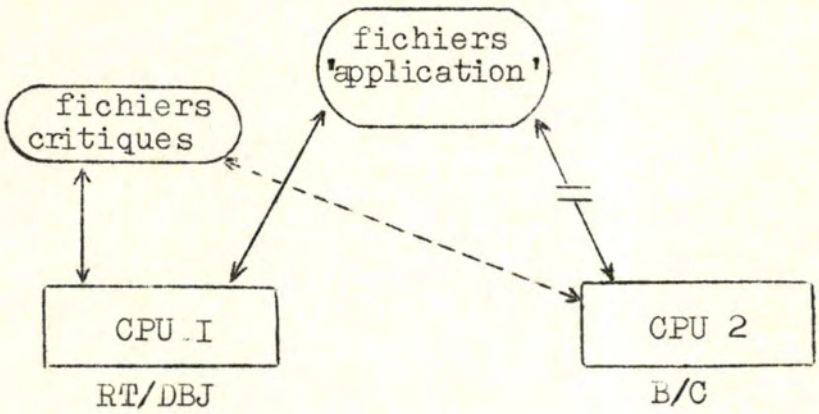
  1) Les fichiers "APPLICATION" sont accessibles aux deux CPU's. Il faut, dès lors, éviter que les deux unités mettent à jour simultanément une même zone d'un fichier. Une "lock list" (voir glossaire), gérée par une des unités, permettrait à un CPU de se réserver momentanément un monopole d'action sur une partie du fichier.

2) Les fichiers "critiques" sont également communs aux deux CPU's. Leur gestion (du point de vue place tambour) sera, de ce fait, exécutée par une des deux unités (soit CPU.1). Si donc CPU.2 effectue une mise à jour d'un enregistrement, il en informera CPU.1 qui exécutera un "beforelook" de l'enregistrement, l'information échangée entre les CPU's consistant en un bloc de contrôle comprenant toutes les informations nécessaires (adresse de l'enregistrement dans le fichier "application", ...). Il en ira de même pour l'exécution d'un "afterlook". Si CPU.1 exécutait un "beforelook" ou un "afterlook" pour le compte de sa propre exploitation, le bloc de contrôle reçu de CPU.2 serait placé dans une file d'attente.

3) Les "afterlooks" restent utiles après le traitement des transactions qui les ont créés (ce qui n'est pas le cas des "beforelooks"; ils sont, de ce fait, conservés sur bande. Seul, un des CPU's se chargera, lorsque le volume de ces "afterlooks" aura atteint un niveau déterminé, de transférer ces enregistrements du tambour vers la bande des "afterlooks", afin de réduire le nombre d'entrées/sorties.

4) Les deux CPU's utilisent la même librairie du système. De plus, le RTOS attribue automatiquement, à chaque run, une place tambour dans laquelle il conserve les éléments de programmes nécessaires pour exécuter les différents jobs du run. Afin que la gestion de cette zone tambour soit sans équivoque, elle sera également effectuée par un seul CPU.

B) Deuxième configuration : B/C - RT/DBJ



- soient le CPU.1 traitant le temps-réel et les travaux en DBJ's.
  le CPU.2 gérant les communications et exécutant les travaux en "batch pur".

- CPU.1 a la possibilité de consulter et de mettre à jour les fichiers "application" cependant que CPU.2 n'a le droit que de les consulter. On pourrait admettre, néanmoins, que CPU.2 puisse mettre à jour les fichiers mais alors, uniquement en dehors des heures d'exploitation du temps-réel.

- Le problème de la "run library" reste identique à la première configuration.

- La gestion des fichiers critiques est facilitée. Un seul de ces fichiers, celui des FH-TAR's (files d'attente des messages d'entrées et de sorties), sera utilisé par l'unité traitant les communications. Tous les autres (AFL's, BFL's, ...) se verront utilisés par le seul CPU traitant les transactions et les DBJ's.

Si aucun choix n'a été effectué jusqu'à présent sur le type de configuration, ce choix portera essentiellement sur la charge que représentent les différentes fonctions (communications, traitement des transactions, DBJ's et "batch pur") à assumer par le système.

2.1.1.2. - Utilisation des périphériques dans un système à deux ordinateurs.

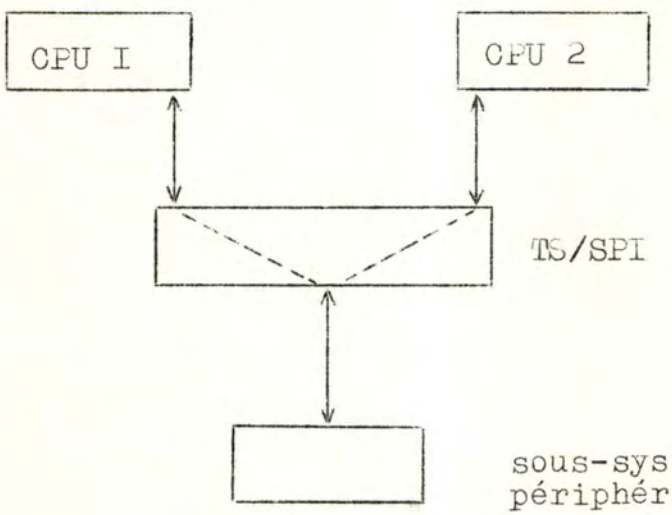
Certaines unités permettent de connecter plusieurs ordinateurs (deux dans notre cas) à un ou plusieurs sous-systèmes périphériques. L'utilisation de ces unités va donc de pair avec le dédoublement de l'unité centrale. Ce sont essentiellement :

- les "Transfert Switch" (TS)
- les "Shared Peripheral Interface" (SPI)

Remarque : on définit un sous-système (UNIVAC) comme un ensemble d'unités périphériques relié à la même unité de contrôle.

A. - Principe

1. Configuration en "Y"



Connection de 2 CPU's à un même sous-système périphérique

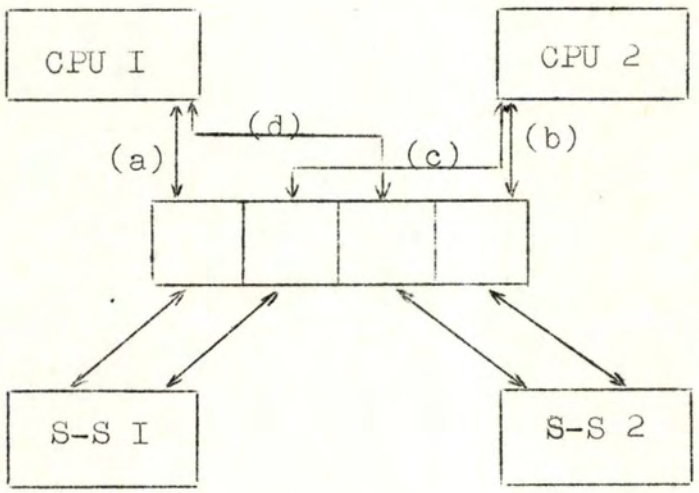
principe d'exclusion : lorsqu'une liaison est réalisée, à un moment donné, entre un CPU et le sous-système périphérique, l'autre CPU n'a plus accès à ce sous-système.

L'accès au sous-système par l'un ou l'autre des CPU's se fera :

- soit par intervention de l'opérateur (cas du TS)
- soit par demande dynamique d'un CPU (cas du SPI).

Remarque.
Dans certain cas, l'unité d'aiguillage (TS ou SPI) est elle-même dédoublée.

2. Configuration en "X"



connection de 2 CPU's à 2 sous-systèmes périphériques.

Deux groupes de liaisons sont possibles :

- un premier groupe de "liaisons" permet à un CPU d'accéder à un seul sous-système périphérique. Si la liaison (a) est réalisée, CPU.2 ne pourra avoir accès qu'à S-S2 s'il en fait la demande.
  Si la liaison (d) est réalisée, seule (c) sera acceptée.
  Désavantage : dans l'éventualité d'une défaillance d'une unité centrale, il n'est pas possible de connecter les deux sous-systèmes périphériques au seul CPU restant.
- un deuxième groupe de "liaisons" permet à un CPU de s'allouer un ou deux sous-systèmes périphériques. A un instant donné, les liaisons (a) et (d) ou (b) et (c) peuvent être réalisées.

Signalons que seul, le premier groupe de liaisons est réalisable à l'ANMC.

B. - Le "Shared Peripheral Interface" (SPI)

Le SPI est utilisé pour connecter deux ordinateurs à un ou plusieurs sous-systèmes périphériques, si l'on désire que les connections se fassent par "accès". En d'autres termes, le sous-système périphérique sera accédé dynamiquement par l'un ou l'autre des CPU's en fonction des demandes d'accès provenant de ces unités centrales. De ce fait, un SPI n'aura sa pleine efficacité que dans le cas où le ou les sous-systèmes périphériques concernés comportent des unités périphériques "partageables" (exemples : les tambours et les disques).

Le SPI se caractérise comme étant une unité essentiellement "active". Les demandes d'accès provenant des deux ordinateurs sont gérées par lui. Il possède, à cet effet, des registres de mémorisation des demandes et des circuits de contrôle de priorités.

Notons également que les liaisons entre le sous-système périphérique et les CPU's s'effectuent de façon entièrement "automatique" sans intervention de l'opérateur ni des CPU's eux-mêmes.

C. - Le "Transfert Switch" (TS)

Le TS alloue un sous-système périphérique à un CPU par "période de travail" et non plus par accès. Un ordinateur se réservera donc un monopole d'action sur un sous-système périphérique pendant toute la durée d'un travail. Pendant cet intervalle de temps, il ne sera pas possible au second CPU d'accéder au sous-système concerné. Le TS ne sera utilisé que dans le cas où le sous-système se compose d'unités périphériques "non partageables" (exemple : les bandes, les imprimantes, les lecteurs de cartes).

Le TS est une unité dite "passive". Il n'a pas à traiter de conflits d'accès contrairement au SPI. De plus, les connections se font "manuellement" par intervention de l'opérateur.

D. - Choix des unités.

Les deux unités (SPI et TS) ont un même objectif : rendre accessible à deux ordinateurs, un ou plusieurs sous-systèmes périphériques. La question qui se pose, dès lors, est la suivante : "Quelle unité va-t-on choisir, étant donné les propriétés des différents types de sous-systèmes et étant donné que la valeur moyenne d'un SPI est trois fois plus élevée que celle d'un TS ?

Malgré qu'il soit possible d'utiliser n'importe quelle unité (SPI ou TS) afin de connecter plusieurs CPU's à n'importe quel type de sous-système périphérique, il serait absurde de connecter deux CPU's à un sous-système "bande" au moyen d'un SPI; un tel sous-système n'est pas "partageable", contrairement aux sous-systèmes "disque" et "tambour". Pour ces deux derniers types de sous-systèmes,

la dépense d'un SPI se justifiera pleinement; elle permettra au système d'acquérir un "throughput" très élevé.

Solution retenue par le Projet :
- SPI pour les sous-systèmes "disque" et "tambour"
- TS pour les sous-systèmes "bande", "lecteurs-perforateurs de cartes", "imprimantes" et le contrôleur des lignes.

2.1.2. - Dédoublement des voies d'accès à un sous-système périphérique.

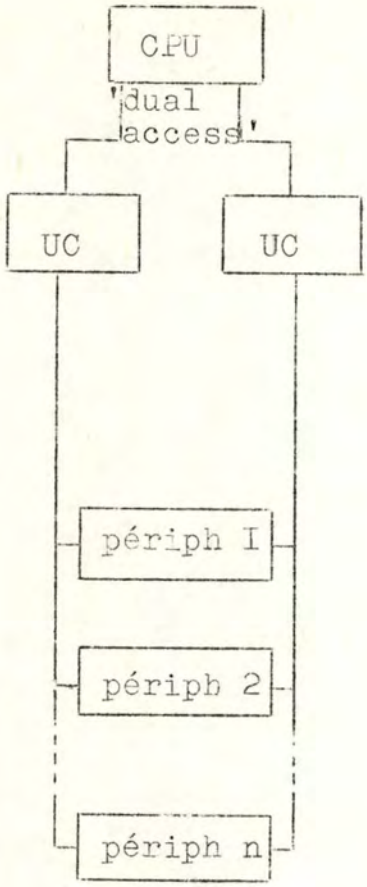
Il est intéressant, dans un système où les informations doivent être accédées très rapidement, d'augmenter les possibilités d'accès aux périphériques. La solution utilisée dans certains cas est celle du "Dual Access".

Après en avoir montré le principe et un cas d'utilisation pratique, nous tenterons d'en justifier l'emploi ou le non-emploi dans les différents sous-systèmes périphériques.

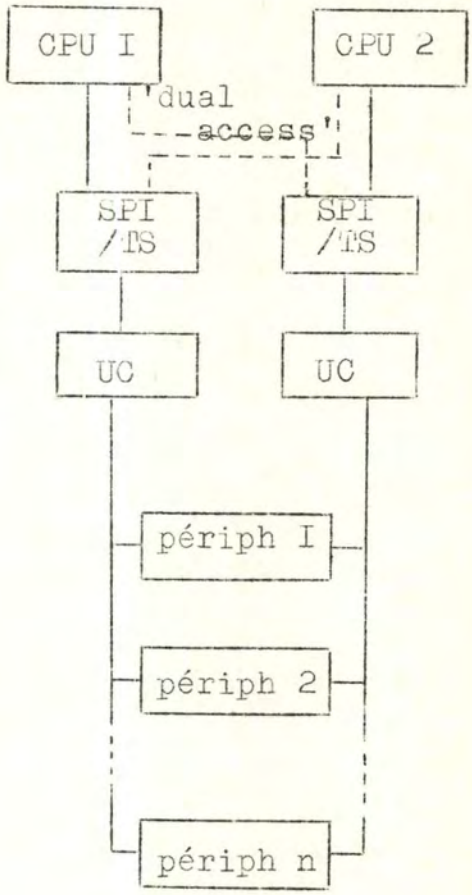
2.1.2.1. - Principe du "Dual Access".

> Le "Dual Access" offre à l'utilisateur la possibilité d'accéder simultanément à un même sous-système périphérique au moyen de deux chemins ("paths") différents, pour autant que ces accès simultanés se fassent vers des unités périphériques différentes.

cas d'un CPU



cas de deux CPU



Il en résulte, pour le système, une augmentation intéressante du "throughput".

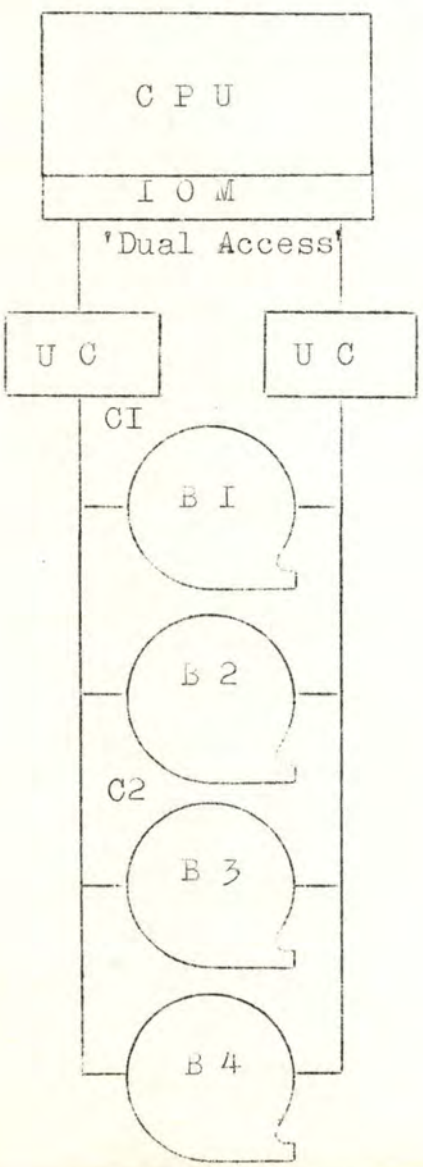
. Le "DUAL ACCESS" exige l'emploi de deux unités de contrôle reliées à l'ordinateur par des canaux différents et d'un certain nombre de circuits supplémentaires situés soit au niveau du périphérique, soit au niveau de l'unité de contrôle. Ces circuits ont pour but de gérer les différents transferts d'informations (ex. : si un périphérique est accédé par un "path", il faut interdire tout accès à cette même unité par l'autre "path"). Il est à noter que ces deux "paths" ne sont pas indépendants. En effet, les demandes d'écriture ou de lecture sont originaires d'une seule et même file d'attente.

. L'avantage considérable qu'offre le "DUAL ACCESS" est qu'il permet, en cas de défaillance d'une unité de contrôle, de conserver le sous-système accessible et d'éviter, de cette façon, une interruption de l'exploitation, ce qui serait catastrophique pour un système en temps-réel.
Il reste à voir si l'avantage que l'on en tire compense la dépense supplémentaire nécessitée. Il en sera question un peu plus loin.

2.1.2.2. - "Duplex file".

Un cas d'utilisation du "Dual Access" est constitué par les "duplex files", c'est-à-dire des fichiers conservés en deux copies. Voyons plus en détail les motifs qui ont poussé à l'emploi du "Dual Access" pour ce type de fichier.

1. Procédé actuel.



. Soient les bandes B1 et B3, les supports des 2 copies du "duplex file".

. I.O.M. (input/output module) = interface d'entrées et sorties.

. Les 4 bandes peuvent être accédées par les 2 unités de contrôle. Une même bande ne peut être accédée simultanément au moyen des 2 "paths".

fig. 2.1.2.2.a : Exemple de "Dual Access" dans un sous-système "bandes".

- opération de lecture : la lecture se fera sur l'unité (B1 ou B3) offrant l'accès le plus rapide.
- opération d'écriture : les fichiers "duplex" sont indispensables à l'exploitation et, de ce fait, ils nécessitent des mesures de protection supplémentaires lors de toute manipulation. Un ordre d'écriture sera dédoublé en deux ordres "hardware" (dédoublement des I/O calls).
  - Il est exigé que ces 2 ordres ne s'exécutent pas simultanément afin de parer à l'éventualité d'une panne de courant ou d'une erreur de l'IOM.
  - Il n'est pas exclu, par contre, que les 2 copies soient écrites (consécutivement) au moyen d'un même "path" (même unité de contrôle).

2. Imperfections du procédé.

Le système décrit au point précédant fait apparaître certaines lacunes importantes :

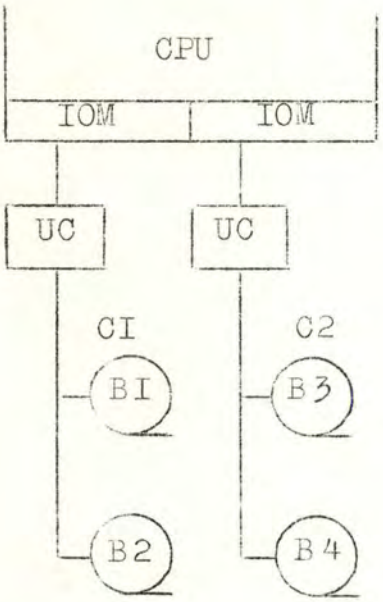
A. - Le procédé n'imposant pas l'utilisation des deux unités de contrôle pour l'écriture des deux copies, une unité de contrôle défaillante peut donc amener l'imperfection des deux copies. Il serait donc utile, dans ce cas, d'obliger l'exécution des deux ordres d'écriture par des "paths" différents.

- L'obligation peut être rendue possible en dédoublant la file d'attente devant les deux unités de contrôle. Mais cela impliquerait une modification du RTOS, ce qui est toujours délicat.

- Une seconde solution serait d'imposer, lors de l'assignation des deux fichiers (2 copies) par l'utilisateur, que ceux-ci se situent dans deux sous-systèmes différents. Cette solution remet en question l'emploi du "Dual Access" dans le cas des "duplex files".

B. - Considérons, à présent, une défection "permanente" au niveau de l'IOM. Il s'ensuit que le ou les sous-

systèmes périphériques dépendants deviennent inutilisables. Une fois encore, la solution serait 2 sous-systèmes indépendants, comme le montre la figure suivante.



. Le 418-III possède 2 IOM's indépendants l'un de l'autre.

. Cette configuration est équivalente à celle de la fig. 2.1.2.2.a. Elle est également moins coûteuse que la première. En effet :

- il y a autant de dérouleurs et d'unités de contrôle.
- il n'y a pas de circuits de "Dual Access".

fig. 2.1.2.2.b. - Deux sous-systèmes périphériques.

3. Choix d'une configuration.

Les deux configurations présentées (fig. 2.1.2.2.a.et t fig. 2.1.2.2.b.) possèdent des avantages et des inconvénients. Il semble néanmoins que la solution du "Dual Access", quoique imparfaite et plus coûteuse, l'emporte sur sa rivale pour deux raisons essentielles :

- la continuité dans l'exploitation est mieux assurée. Dans la seconde solution, la défaillance d'une unité de contrôle rendrait inaccessible une des deux copies, ce qui ne serait pas le cas avec le "Dual Access"
- les défaillances de l'IOM sont très rares.

2.1.2.3. - Opportunité du "Dual Access" dans les différents sous-systèmes périphériques.

Il s'agit ici d'apprécier l'emploi du "Dual Access" en fonction des différents types de sous-systèmes périphériques.

1. Sous-système "DISQUE".

Les disques, dans l'entreprise, supportent les fichiers "APPLICATION" (c'est-à-dire les données des utilisateurs). Les unités employées à cet effet sont les UNIVAC - 8640 dont la capacité est de 176.160.768 mots de 18 bits. Seules deux unités de ce type peuvent être raccrochées à la même unité de contrôle et former ainsi un sous-système. Cette limitation est uniquement due à une insuffisance d'adressage, les unités étant très volumineuses. Ajouter à cela le fait que de tels sous-systèmes ne supportent pas de "duplex file", il est clair qu'un "Dual Access" ne se justifie nullement.

2. Sous-systèmes "BANDE" et "TAMBOUR".

Ces deux types de sous-systèmes périphériques contiennent des "duplex files". De plus, 16 unités peuvent composer, au maximum, un sous-système "bande"; 8 unités dans le cas d'un sous-système "tambour". Le "Dual Access" sera efficace dans les deux cas.

Remarque.

Certaines unités périphériques, telles que les disques et les tambours, sont très sensibles à la température.

Il a été observé, à l'ANMC, que certaines défectuosités dans le conditionnement d'air, apparaissaient par suite d'un manque d'adaptation de certaines fonctions utilitaires (exemple : portes mal adaptées, ...)

.....

Section 2. - Mesures "Software".

2.2.1. - Mesures relatives au traitement interne d'une transaction.

Il faut entendre, par "traitement interne d'une transaction", l'exécution, en mémoire centrale, des différentes phases du RTS.

On considère donc, dans ce paragraphe, la succession des pages "système" et "application" en négligeant toutefois les manipulations avec les fichiers. Celles-ci et les problèmes les concernant, feront l'objet du paragraphe suivant.

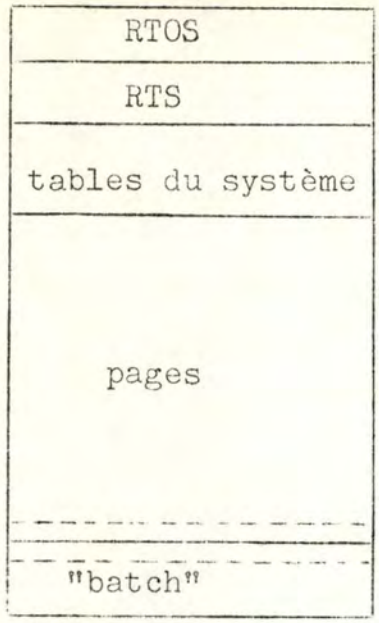
Il sera question ici essentiellement de protéger le système contre les activités "illicites" des transactions.
Retenons, principalement, les mesures concernant :

- la protection "mémoire"
- et l'exécution correcte des pages en mémoire.

2.2.1.1. - La Protection "mémoire".

Cette protection consiste en une protection d'écriture. Elle est indispensable afin qu'un programme ne puisse écrire des données en dehors de la zone-mémoire qui lui est allouée. Le RTOS prévoit, à cet effet, l'utilisation du "guard mode". Ce procédé consiste à charger, dans un registre spécial, les adresses supérieure et inférieure déterminant la zone "opérationnelle", en mémoire centrale, d'un programme. C'est ainsi qu'avant chaque transfert en mémoire centrale, il sera effectué un contrôle s'assurant que l'adresse de destination est comprise entre les valeurs limites contenues dans le registre. Dans le cas où le résultat du contrôle serait négatif, une interruption de violation du "guard mode" serait générée et le contrôle transféré à la routine appropriée du superviseur.

A) Zones opérationnelles.



configuration de la mémoire centrale.

. Le RTOS et le RTS s'exécutent en mode "ouvert" c'est-à-dire qu'ils ont la possibilité de modifier n'importe quelle zone de mémoire centrale (pas de "guard mode").

. Les pages :

- pages "application" : elles ont le droit de modifier le contenu des TAR's pour lesquelles elles se déroulent. Le registre du "guard mode" contiendra donc les adresses de début et de fin de la TAR pour laquelle s'exécute la page "application".
- pages "système" : les pages exécutent des transferts de données dans la zone de mémoire centrale réservée aux tables du système et dans les pages contenant les blocs de contrôle (les "packets"). Leur "guard mode" s'étend sur toute la zone des pages et sur les tables du système.

B) Faiblesses du procédé.

1. Les pages "système" ne sont soumises que partiellement au principe du "guard mode". De ce fait, elles ont la possibilité de détruire la zone allouée aux "pages" de même que les tables du système.

2. Le programmeur peut, à sa demande, ne pas utiliser le "guard mode". Le cas n'est possible que pour les travaux "batch". Il est parfois difficile, pour certains de ces programmes, de définir quelle sera leur zone opérationnelle (exemple : un programme de tri), mais il serait tout de même utile, voire indispensable, de protéger la partie résidente du système contre toutes erreurs d'adressage émanant de ces programmes.
La protection par "guard mode" ne peut pas être efficace s'il existe la possibilité d'y passer outre. On pourrait exiger que les travaux "batch" s'imposent une zone opérationnelle maximum, même si celle-ci se révèle trop grande à l'exécution.

3. L'existence d'un seul registre de "guard mode" pose également un problème lorsqu'il faut contrôler simultanément plusieurs zones de mémoire centrale.
Une transaction a la possibilité, si le besoin s'en fait sentir, de s'adjoindre plusieurs zones de travail supplémentaires ("data page") en plus de sa propre TAR. Le "guard mode" étant fixé sur cette TAR, il est donc impossible d'exécuter des instructions de transfert vers ces zones de travail. Il a donc été nécessaire, dans ces cas, d'exécuter une routine intermédiaire (appel "superviseur") dans laquelle un certain nombre de contrôles sont effectués afin de déterminer notamment si la "data page" appartient bien à cette transaction. Il s'agit là d'une procédure assez lourde à l'exécution.

C) Améliorations possibles.

1. Imposer que les tables du système et les pages "système" occupent une zone définie en mémoire.

Le registre de "guard mode", lors de l'exécution d'une page "système", serait fixé sur cette zone.

2. Introduction d'un second registre.

Ce registre ne contiendrait qu'un seul bit. Si ce bit est "on", la protection s'exercerait à l'intérieur du segment délimité par les adresses supérieure et inférieure contenues dans le premier registre. Si ce bit est "off", la protection s'exercerait à l'extérieur de ce segment.
Ce deuxième registre permettrait de protéger certaines zones de mémoire lorsqu'il n'est pas possible de déterminer quelles seront les limites opérationnelles d'un programme.
Cette solution est purement théorique et exigerait une modification du "hardware" utilisé.

3. Plusieurs paires d'adresses supérieure et inférieure pourraient être utilisées dans le cas d'un contrôle simultané de plusieurs zones.

Mais si l'on cherche à généraliser cette solution à un nombre quelconque de zones de mémoire, le coût de la solution deviendrait excessif et l'exécution des instructions en serait considérablement ralentie.

4. D'autres solutions n'utilisant pas le procédé du "guard mode" sont possibles. Citons principalement la protection par clés (adoptée par IBM 360). Cette solution repose sur une division de la mémoire en blocs fixes, chacun de ceux-ci possédant une clé de protection.

A tout programme est également associée une clé de protection. Lorsqu'une instruction du programme fait référence à une adresse, une comparaison est effectuée entre la clé de protection du programme et celle associée au bloc contenant cette adresse.

5. Imposer le respect de certaines conventions par l'utilisateur.

Les pages "système" et "application" sont implémentées sur tambours dans une zone qui leur est propre. Il ne

peut être question, pour un utilisateur, d'implémenter son propre programme dans la zone des pages "système" si ce programme ne constitue pas une fonction système.

Exemple : un programme de facturation, bien testé, et qui, de l'avis du programmeur responsable, ne nécessiterait aucune modification ultérieure, fut implémenté dans la zone réservée aux programmes "système". Or, ce programme faisant appel à de nombreux textes de lois, exigea, en réalité, de nombreuses modifications. Il s'ensuivit qu'après une de ces modifications, le programme écrasa les informations contenues dans les pages voisines.

2.2.1.2. - Exécution correcte des pages en mémoire.

Les pages "software" sont, en quelque sorte, l'élément dynamique du système. Ce sont elles qui lancent les différentes activités d'exécution telles que les opérations d'entrées et sorties, les appels "superviseur" et bien d'autres encore. Mais elles sont, malheureusement, l'origine de la plupart des erreurs ou destructions d'informations.

Il est donc important d'en assurer un contrôle strict tant au niveau de la "vraisemblance" (certaines instructions "privilégiées" ne peuvent s'exécuter que dans les pages "système") qu'au niveau de la "validité" (cet appel superviseur contient-il tous les paramètres requis ?) des éléments qui les composent.

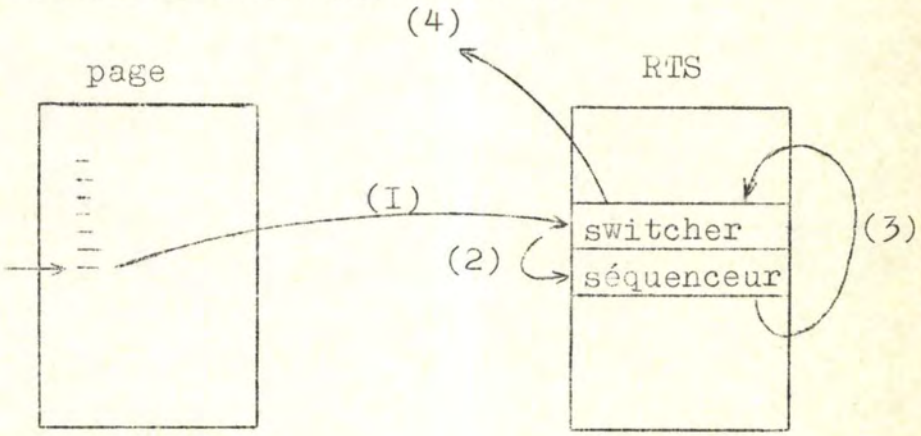
A) Les appels "superviseur".

Le RTS met à la disposition de l'utilisateur, un certain nombre d'appels "superviseur". Le mot "superviseur" est, sans doute, mal choisi. Il s'agit, en fait, d'appels aux routines du RTS. Celui-ci offre un certain nombre de facilités telles que :

- le gestionnaire des pages "hardware";
- les "handlers" c'est-à-dire les routines dont le but est l'interception par le RTS des opérations d'entrées/sorties;
- les routines de gestion :
  - . des "afterlooks"
  - . des "beforelooks"
  - . de la "lock list"
- certaines routines de traitement des erreurs (ABORT).

Les pages "SYSTEME" ont la possibilité d'exécuter n'importe quel appel "superviseur". Il n'en va pas de même pour les pages "APPLICATION". L'idée suivie est que moins on laisse de possibilités à l'utilisateur (c'est-à-dire aux pages "APPLICATION"), moins ce dernier fera d'erreurs. Le principe des "handlers", qui sera développé plus tard, est un exemple de cette philosophie.

Exécution d'un appel "superviseur".



. Un appel "superviseur" signifie un retour au RTS. Le "switcher" du RTS prend le contrôle (I) afin d'enregistrer l'environnement de la page interrompue. Ce travail consiste, en fait, à ajouter un bloc de contrôle dans une des files d'attente du "switcher".

- Le contrôle passe ensuite au séquenceur (II). Il ajoute un bloc de contrôle à la file d'attente traitant le type d'appel en question. Il exécute alors, une à une, toutes les demandes se trouvant dans les files d'attente des modules "séquenceur".
- En fin de travail du séquenceur, le contrôle est rendu au "switcher". (III)
- Il appartient au "switcher" de choisir quelle sera la page suivante à exécuter. Cette page sera celle dont la priorité est la plus élevée parmi les trois files d'attente du "switcher".

Afin de renforcer davantage la sécurité interne, des contrôles supplémentaires sont exécutés pour chaque appel "superviseur". Voyons quelques exemples :

"Rel, Pag" — Cet appel a pour but la désallocation d'une page d'un utilisateur.
Contrôle : la page à désallouer appartient-elle à l'utilisateur ?

"CALS vpage, vline" — Appel pour une page virtuelle dont l'adresse est "vpage, vline".
Contrôles : cette page virtuelle existe-t-elle ?
l'utilisateur a-t-il le droit d'accéder à cette page ?

Pour tous les appels "superviseur", un contrôle sera également effectué afin de déterminer si l'appel en question provient bien d'une page qui était en activité à ce moment-là. On détecte, de cette façon, si l'utilisateur n'a pas fait un saut ("jump") hors d'une de ses pages, auquel cas il serait en infraction.

Appréciation.

De ce qui vient d'être dit, on peut déduire que :

- la structure logique d'un fichier ne pourra être modifiée par l'exécution d'une page "application". Les erreurs concernant cette structure ne peuvent apparaître que dans les pages "système" ou dans les routines du RTS.

- la gestion des fichiers critiques est uniquement du ressort des pages "système".
  exemples d'appels qui ne peuvent apparaître que dans les pages "système" :

  . ADD, BFL ..... ajoute d'un "beforelock" au fichier BFL.

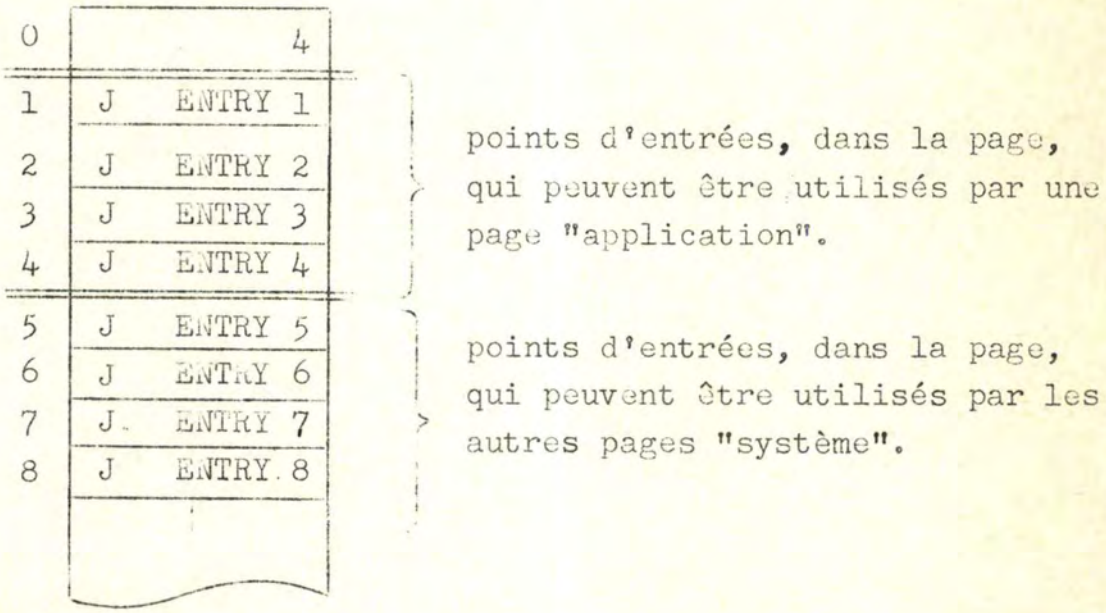
  . SCH, QIN ..... recherche, dans l'input-queue, du message suivant à exécuter.

  . ADD, LCK ..... réservation ("lock") d'une partie du fichier "application".

B) Validité des appels entre pages.

De façon à prévenir tout appel illégal d'une page "application" vers une page "système", les mesures de protection suivantes ont été prises.

Toutes les pages "système" auront le format suivant :



Tous ces mots, excepté le premier, sont des adresses, des "jumps" vers les différents points d'entrées dans le programme.

Le premier mot contient le nombre de points d'entrées possibles pour une page "application". Un premier contrôle est effectué sur ce premier mot; s'il est "o", alors l'appel est refusé. On fait un "ABORT" de la transaction fautive. Si ce premier mot est différent de "o", on compare l'adresse qui accompagne l'appel avec chacun des "n" mots qui suivent le premier ("n" est le nombre contenu dans le premier mot).

Réflexions.

- Le procédé n'interdit pas à une page "application", de faire appel à n'importe quelle page "système", à condition que les points d'entrées réparés soient exacts, c'est-à-dire acceptés par la table des points d'entrées pour les pages "application", comme décrit ci-avant.
- Le format n'est valable que pour les pages "système". On pourrait imaginer de l'étendre aux pages "application". Cela n'a pas été réalisé par le Projet étant donné la multitude de ces pages "application" et vu la réorganisation que nécessiterait une telle implémentation.
  Notons une fois encore que la sécurité est un ensemble de mesures que l'on décide d'appliquer dès la mise en oeuvre d'un système informatique.
- Des systèmes plus complexes peuvent s'imaginer. L'emploi d'une table intermédiaire d'autorisation est souvent proposé dans de nombreux ouvrages.
  Un modèle de table d'autorisation possible est le suivant :

n°page	bits d'autorisation
	(un bit 'on' par page accessible)

table d'autorisation des pages

A chaque numéro d'identification d'une page "application" ou "système" correspond la liste (bits "ON") des pages accessibles par cette page. Ainsi, à chaque appel d'une page, la table sera inspectée dans une première étape.

Si ce premier test s'avère positif, un second test sera effectué dans la page appelée, sur les points d'entrées possibles de cette page et cela, de la façon décrite précédemment.

2.2.1.3. - Les programmes statistiques.

Les programmes statistiques sont exécutés à partir des enregistrements du "logging". Les résultats de ces programmes peuvent avoir une importance capitale pour la suite de l'exploitation.

L'analyse statistique portera notamment sur :
- le nombre d'appels entre pages
- le nombre d'entrées/sorties par interaction
- le temps moyen d'une opération d'entrée/sortie
- le nombre d'appels pour chaque type de transaction
- le nombre de "clean-up" et leur distribution pendant la journée

- le nombre d' "ABORT" par journée
- le nombre d' "afterlooks" et de "beforelooks" par journée.

En fonction des résultats obtenus, certaines décisions seront prises concernant, entre autres :
- l'ordonnancement des travaux en différé
- la modification de certaines routines d'exécution
- l'ajoute de certaines unités périphériques.

2.2.1.4. - Deux mesures préventives.

A. "TRANSACTION PACKET"

L'utilisateur étant capable de détruire sa propre TAR, il est utile de conserver les informations critiques représentant l'état instantané de la transaction (adresse de la page virtuelle en cours d'exécution, numéro du message, adresse du dernier "beforelook", adresse de la TAR, ...)

A chaque "transaction starter", ces informations seront stockées dans une zone spéciale ("transaction packet") à l'abri de toutes les manipulations de l'utilisateur; seul, le système a le droit de mettre à jour ces informations et de les restaurer dans la TAR au moment du "clean-up".

B. "CRITICAL AREA"

La "critical area" contient des informations sur l'état du système, notamment :
- le nombre de transactions actives à un instant donné
- le numéro du dernier message traité
- le nombre de DBJ's actifs
- certaines informations concernant l'état des fichiers "système"
- les données concernant les tables du système (NDT : Network Description Table - LDT : Line Description Table - TDT : Terminal Description Table)

L'utilisation de la "critical area" est très importante dans les procédures de recouvrement. Il en sera question dans le chapitre consacré aux procédures de secours.

2.2.2. - Mesures relatives à l'exploitation des fichiers.

Les mesures analysées dans cette partie concernent les manipulations des transactions avec les fichiers "application". Ces mesures visent à assurer l'intégrité des informations lors de ces manipulations.

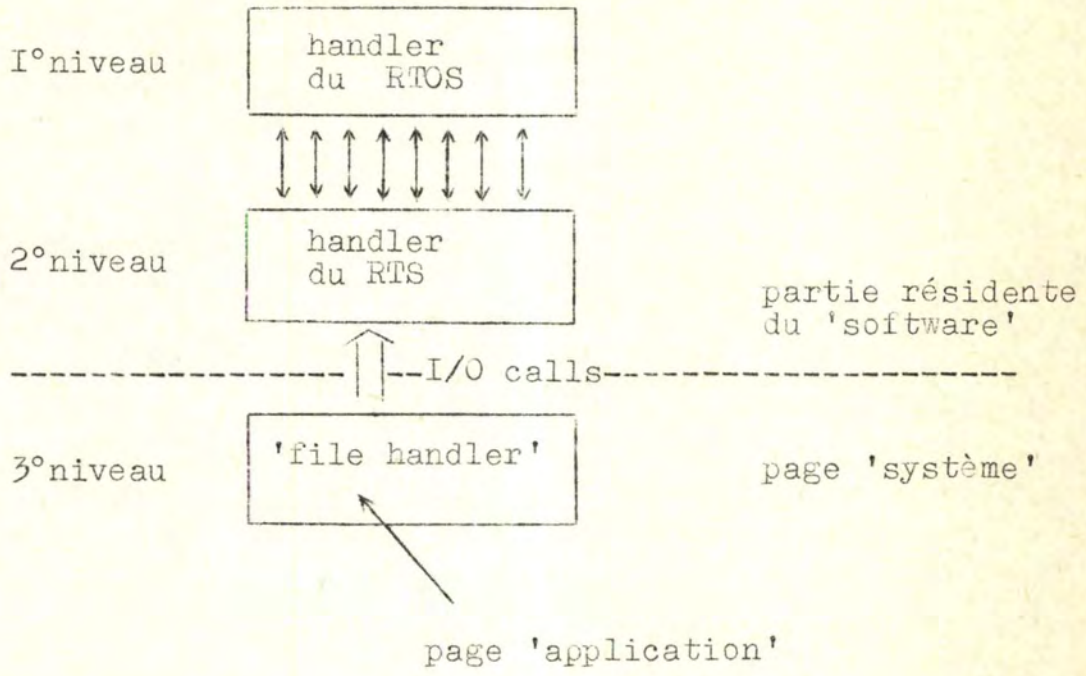
2.2.2.1. - Les "handlers".

Le RTS inclut tout un ensemble de "file handlers" qui répondent aux besoins fondamentaux d'un système de contrôle des transactions. Les "file handlers" se constituent d'une collection de routines qui permettent aux utilisateurs aussi bien qu'au RTS lui-même de manipuler les données sur fichier sans jamais être en contact direct avec les fichiers concernés.

Le "file handler" a été développé en tenant compte des normes suivantes :

- chercher un compromis entre les facteurs temps d'exploitation et place occupée par les "handlers" en mémoire;
- offrir des facilités pour la consultation (optimisation du nombre d'accès);
- assurer un haut niveau de sécurité.

A. - Fonctionnement.



- Le RTOS possède un certain nombre de routines dont le but est le transfert physique des informations entre la mémoire et le périphérique. C'est à ce niveau qu'est effectuée la conversion de l'adresse logique en adresse physique. Il s'agit du niveau le plus bas du contrôle.

- Le RTS possède également des "handlers". Ils constituent en fait, une extension du RTOS. Cette extension est nécessaire pour informer le RTOS de certaines spéficications, purement techniques, concernant le type de l'unité périphérique en question.
  Exemple : si un sous-système "bande" utilise deux unités de contrôle ("Dual Access), c'est à ce niveau qu'il en sera tenu compte. Le "handler" du RTS connaît également le type d'un fichier ("duplex file", "simplex file", fichier "read only", ...)

. Le troisième niveau, celui directement accessible par la page "application", est constitué du "file handler". Le "file handler" est une page "système" dont le but est de tenir compte, lors d'une opération d'E/S, de la structure des enregistrements du fichier. Les fichiers "application" ont une organisation particulière (organisation en "tiroirs") dont l'existence est ignorée par l'utilisateur.

. Les manipulations sur les fichiers sont lancées par un appel émanant de la page "application" et destiné "au file handler" concerné mais le contrôle de ces manipulations incombe entièrement au système.

B. - Intérêt des "handlers".

. L'emploi des "handlers" simplifie l'effort de programmation. Il y a un seul "handler" pour toutes les opérations d'E/S sur un même type de périphérique.

. L'emploi des "file handlers" s'est avéré indispensable étant donné la complexité de l'organisation des fichiers "application".

. La non-duplication du codage des entrées/sorties dans chaque programme permet un gain considérable de place mémoire. Il s'agit d'un avantage précieux pour tout système en multiprogrammation.

. Les "handlers" rendent impossible toute modification de la structure d'un fichier "application" par l'utilisateur (les TPS's). Une destruction de cette structure n'aura son origine que dans le codage du "file handler".

2.2.2.2. - Réservation des fichiers "APPLICATION".

On entend par "réservation du fichier", la possibilité qu'a le programmeur de se réserver un monopole d'action sur une portion du fichier et cela, pendant un certain temps. Il s'agit de protéger un utilisateur, travaillant sur une partie du fichier, contre les appels d'autres utilisateurs concernant la même portion du fichier. Pour ce faire, le système tient à jour un fichier des réservations (fichier des "locks") contenant toutes les réservations faites à un instant donné.

A. - Utilisation de la "lock list".

Lorsque l'on désire mettre à jour un enregistrement sur fichier, le "file handler" exécute un "ADD-LCK". Cet appel superviseur a pour but d'analyser le fichier des réservations afin d'analyser si la zone demandée est réservée ou non.

- Si NON, le système génère un "lock" (enregistrement de six mots).

1	pointeur
2	adresse de début
3	
4	adresse de fin
5	
6	n° trans.

. Les "locks" sont chaînés suivant les adresses croissantes dans le fichier. Pour une demande de réservation, il suffira de "scanner" le fichier jusqu'au "lock" ayant une adresse de début supérieure à celle de la zone demandée.

. Les adresses de début et de fin déterminent la zone qui fait l'objet d'une réservation.

. Le 6e mot identifie la transaction.

- Si OUI, une partie ou la totalité de la zone a déjà fait l'objet d'une réservation (il est tenu compte des chevauchements éventuels). La transaction qui a fait la demande de réservation est placée dans une file d'attente.

  Chaque fois qu'un "lock" est supprimé (au "clean-up" d'une transaction), la file d'attente est analysée afin de détecter si une demande de réservation n'a pas été faite concernant la même partie du fichier.

B. - Appréciation.

Le procédé de la "lock list" est interne au RTS. Les programmes se déroulant sous le RTOS n'en usent pas alors qu'ils ont la possibilité de consulter et de mettre à jour les fichiers "application".

Cette dualité d'accès aux fichiers est un problème crucial. Des propositions seront avancées en section 3 de ce chapitre.

2.2.2.3. - Contrôle des "labels".

Le contrôle des "labels" est un moyen de protection efficace contre les erreurs humaines, notamment des opérateurs ou du gestionnaire des bandes, s'il existe. C'est un contrôle indispensable et il est difficile de s'imaginer qu'un centre puisse se passer d'une mesure aussi nécessaire et, en particulier, dans le cadre d'un système en temps-réel.

Il existe, dans le RTOS, une routine (VOLAB$) dont le but est d'écrire, sur bande, un label de volume et un label de fichier ("header label") :

- le label de volume contient, en outre, un numéro sériel de bande. Un tel numéro doit être unique et fixe pour chaque bande.
- le label de fichier contient, notamment, une date de création et une date d'expiration.

La logique imposerait qu'à chaque nouvelle utilisation d'une bande, un contrôle soit effectué sur ces deux labels afin de s'assurer que la bonne bande ait été choisie (d'après le numéro sériel et la date de création) et que celle-ci ait une date de péremption antérieure à la date du jour.

En fait, aucun de ces contrôles n'est effectué. Dès qu'une bande est utilisée, la routine VOLAB$ est exécutée systématiquement et cela, sans se soucier du contenu ancien des deux labels. Il y a deux raisons essentielles :

1. Si certaines bandes ont un label de volume et un label de fichier (c'est le cas des bandes utilisées pour le temps-réel), d'autres, par contre, n'en possèdent pas (certaines bandes de travail).
   De plus, il n'a pas été attribué de numéro sériel pour chaque bande. De ce fait, il n'est pas possible de différencier les bandes qui possèdent des labels de celles qui n'en possèdent pas. L'opérateur choisissant une bande dans la bibliothèque, ne se pose pas la question de savoir si cette bande possède ou non un label de volume et un label de fichier; il demande l'exécution de VOLAB$ qui lui accorde une sorte de "permission" d'utilisation.
2. Lorsque l'on exécute VOLAB$, la date d'expiration que l'on introduit est de 90 jours consécutifs à la date de création (date du jour). Or, le taux d'utilisation d'une bande est actuellement de 8 jours. Donc, si le contrôle de label s'effectuait réellement, la plupart des bandes seraient refusées soit parce qu'elles ne possèdent pas de label, soit parce que leur date d'expiration est postérieure à la date du jour. C'est une deuxième raison pour laquelle l'opérateur exécute VOLAB$ sans se soucier du contenu ancien du label.

Améliorations possibles.

1. Il semble qu'une première mesure consisterait à attribuer un numéro sériel à chacune des bandes, du moins à celles participant au temps-réel. Dans ce cas, une séparation complète serait effectuée entre les bandes contenant des labels et celles qui n'en possèderaient pas.

2. Avant toute utilisation d'une bande, il serait utile de sortir, sur imprimante, le contenu ancien des deux labels. Il appartiendrait alors à l'opérateur de décider de l'utilisation de cette bande.

3. Il serait utile également d'adapter la date de péremption à la fréquence d'utilisation des bandes ou, dans le cas où ce serait impossible à réaliser, d'augmenter le nombre de bandes afin de diminuer la fréquence d'utilisation.

4. Au départ de chaque exploitation du temps-réel, on spécifierait le numéro de la bande qui contiendrait les enregistrements d'un fichier critique. Un contrôle serait effectué à chaque changement de bande afin de s'assurer que la nouvelle bande ait un numéro sériel supérieur à la précédente.
   On éviterait, de cette manière, les mésaventures du genre :

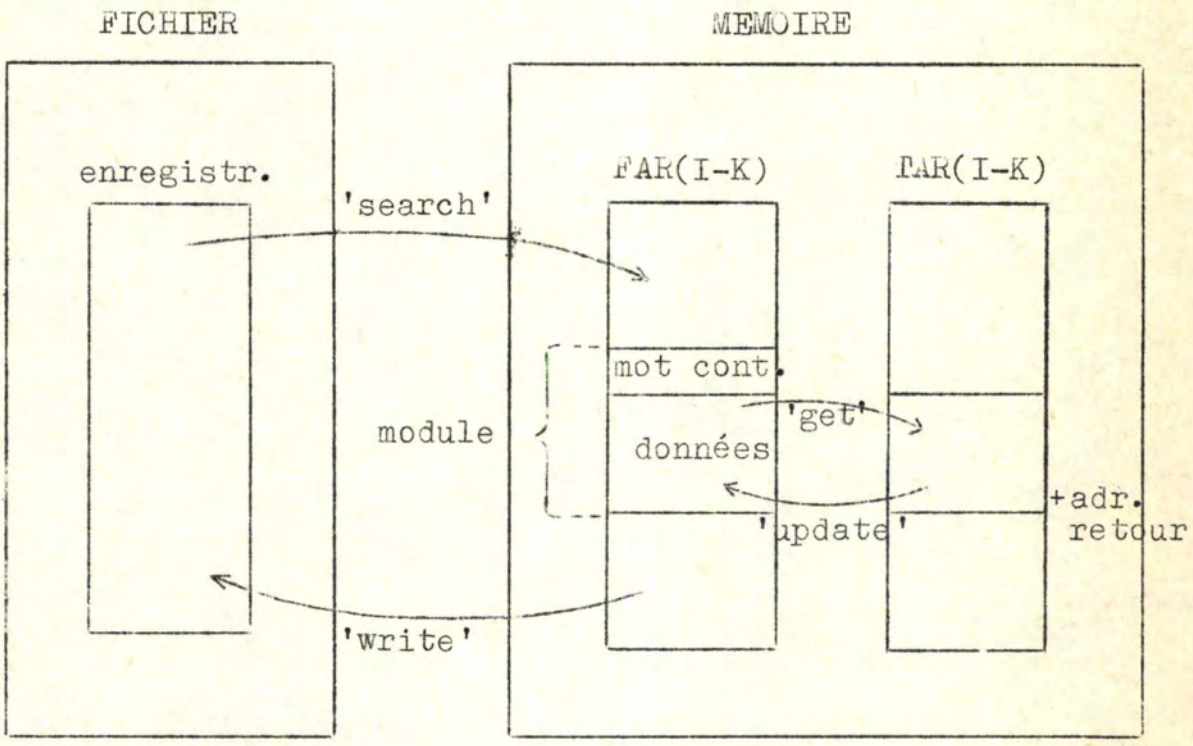
   > lors du chargement d'une nouvelle bande "logging", la précédente étant remplie, l'opérateur - supposant vierge cette nouvelle bande - y plaça, en fait, une bande "logging" de l'exploitation en cours, vieille seulement de quelques heures.

2.2.2.4. - Règles de modifications d'un fichier "application".

Les différentes étapes d'une mise à jour d'un enregistrement du fichier seront envisagées.

L'efficacité des contrôles exécutés lors de toute manipulation sera testée; certaines lacunes de la procédure seront présentées.

A. Etapes d'une mise à jour.



- fonctions de l'utilisateur.

  "SEARCH" - "GET" - "UPDATE" - "WRITE"

- FAR ("File AReа")

  Zone de travail associée à une transaction dans le but de recevoir un enregistrement du fichier. La FAR n'est pas accessible à l'utilisateur.

Première étape : "SEARCH 'enregistrement'"

Cette fonction a pour but de charger, dans la FAR de l'utilisateur, l'enregistrement choisi. Celui-ci est chargé avec tous les modules qu'il comporte. Chacun de ces modules contient des données de contrôle, notamment la date de la dernière mise à jour de même que le type de TPS ayant exécuté cette mise à jour.

Deuxième étape : "GET 'module'"

Seul, le module dont la mise à jour est demandée, sera transféré dans la TAR de l'utilisateur. La raison de ce nouveau transfert découle du "guard mode", celui-ci étant fixé sur la TAR de la transaction.

Les données de contrôle ne sont pas transférées dans la TAR mais elles seront modifiées après la mise à jour du module.

La TAR reçoit également l'adresse de retour du module dans la FAR.

Troisième étape : "UPDATE 'module'"

Une fois le module mis à jour dans la TAR, il est replacé dans son emplacement dans la FAR d'après l'adresse retour. La modification des données de contrôle du module s'effectue à ce moment.

Quatrième étape : "WRITE 'enregistrement'"

Cette étape effectue le transfert du contenu de la FAR vers le fichier.

B. Mesures et contrôles.

. La réservation ("lock list") de l'enregistrement du fichier est supposée réalisée.

- Les données de contrôle associées à chaque module ont une importance capitale en cas d'erreur. En effet, si une erreur est détectée dans le contenu d'un module, le programme fautif sera immédiatement décelé.

- Dans un programme, à tout "UPDATE" doit correspondre un "GET" du même module. Le contrôle s'effectuera, en fait, sur les longueurs respectives des deux modules. Celles-ci doivent être identiques. La mesure n'est pas absolue car plusieurs types de modules peuvent avoir la même longueur.

- Avant d'exécuter un "UPDATE", il est contrôlé si, à l'adresse retour (dans la FAR), correspond réellement un module du même type que celui se trouvant dans la TAR.

- Lorsque l'utilisateur demande l'exécution d'une fonction "GET", il joint à cet appel l'adresse d'une FAR. Le nombre de FAR est limité à 8 par utilisateur. Les adresses de ces 8 zones sont reprises dans le "TRANSACTION PACKET" de l'utilisateur. Lors d'une fonction "GET", la validité de l'adresse de la FAR est contrôlée.

C. Conclusions.

- La fonction "SEARCH" transfert l'intégralité d'un enregistrement. Or, certains de ceux-ci peuvent atteindre des tailles très importantes (le nombre de FAR's est limité à 8).
  Un tel enregistrement monopolise beaucoup de place mémoire et influence, de ce fait, le "throughput" du système. Il serait préférable, dans ce cas, de permettre le transfert de blocs d'information plus petits (le module, par exemple). Certains enregistrements possèdent un nombre élevé de modules du même type, ces modules s'ajoutant au fur et à mesure du temps, de sorte que les premiers, étant donné leur ancienneté, n'ont plus d'intérêt vis-à-vis de l'exploitation.

Une réorganisation du fichier pourrait se faire sur cette base afin de supprimer les modules anciens. Il en résulterait un allègement des enregistrements et, par conséquent, des transferts entre le fichier et la TAR.

Remarque.
Dans certains cas, un chaînage "inverse" est utilisé de façon à accéder d'abord aux modules les plus récents.

- Le contenu, proprement dit, des modules d'un fichier fait rarement l'objet d'un contrôle. En fait, il n'existe aucun programme dont le but est de vérifier l'exactitude des données contenues dans ces modules. Seuls, certains contrôles sont effectués lors d'une application particulière.

  Exemple : au moment de la facturation (une fois par mois), l'ensemble du fichier est parcouru et l'exactitude de certains totaux, dates, ... est analysée.

  Le contrôle de la vraisemblance et de la validité des données se limite à ces mesures occasionnelles.

2.2.3. - Les procédures de secours.

Les procédures de secours revêtent une importance particulière dans un système en temps-réel. Le but d'un tel système étant d'assurer, aux utilisateurs, un service ininterrompu, la règle fondamentale sera de poursuivre l'exploitation dans la mesure du possible, peut-être même en service réduit. Une interruption complète du système doit être considérée comme un cas de nécessité absolue lorsque tout autre moyen s'avère impossible.

Différents types d'erreurs ou de mauvais fonctionnement peuvent se produire. Ces événements, en général,

aboutiront à trois types de mesures :

1. une erreur qui intervient dans le traitement d'une transaction (erreur dans une page "application"), résultera dans l'élimination de la transaction qui l'a provoquée. Le système doit avoir la possibilité de détruire tous les changements provoqués par cette transaction sur les fichiers "application" et de continuer le traitement des autres transactions.
2. une erreur irrémédiable qui aura, comme conséquence, l'interruption du run RTS, exigera que soit réinitialisé le système sans perdre les informations des autres transactions en cours.
3. une erreur dont la conséquence (catastrophique) serait une interruption du run RTS suivi de sa réinitialisation, s'accompagnera de la perte de toutes les informations concernant les transactions actives au moment de l'interruption.

Notons que l'accès des fichiers "application" peut également donner lieu à des erreurs, ce qui provoquera également l'interruption de la transaction concernée.

Toute "recovery" s'accompagnera généralement de deux volets :

1. la reconstitution du fichier, afin de replacer celui-ci dans la situation antérieure à l'interruption. Il s'agit en fait, d'éliminer toutes les modifications introduites dans le fichier par le ou les transactions fautives.
2. la "recovery" des transactions. C'est le problème de la continuité interne des opérations.

REMARQUES IMPORTANTES.

A. Dans le domaine des procédures de secours, le degré selon lequel un système est efficace est fonction du prix que l'on veut bien payer.

Il y a donc un choix à faire entre, d'une part, le coût du matériel et l'effort de programmation et, d'autre part, le prix d'une immobilisation totale ou partielle du système.

B. Il est souhaitable que la détection de l'erreur coïncide avec le moment exact de l'apparition de l'erreur. La détection de l'erreur est essentielle en temps-réel. En effet, une mise à jour erronée, non détectée, peut causer à son tour de nouvelles erreurs indécelables. Insistons donc sur l'importance capitale des tests du programme.

La "recovery" élaborée par le Projet sera analysée en trois étapes. Une première étape établira le type des erreurs provoquant la procédure de "recovery"; une seconde en développera le fonctionnement tandis qu'une dernière tentera de mettre en évidence un certain nombre de problèmes relatifs à la mise au point de la procédure.

2.2.3.1. - Types d'erreur.

A. L'erreur de programme.

Une erreur de programme peut influencer l'exécution d'une seule transaction ou de toutes les transactions actives. Généralement, les erreurs intervenant dans les pages "système", les pages "processeur" ou dans la partie résidente du RTS, seront à l'origine d'une "recovery" de toutes les transactions; les erreurs apparaissant dans les pages "application" n'intéresseront que la transaction concernée.

Les erreurs de programme sont souvent causées par des combinaisons imprévues d'événements; ces combinaisons surviennent très rarement et n'ont, généralement, pas été envisagées au cours des tests du programme. De même,

une modification apportée à un programme peut entraîner des erreurs imprévues.
A titre d'exemple : une procédure valablement testée, dans le cas d'un système exécutant en parallèle huit transactions, peut se trouver erronée dès que le système accepte neuf transactions ou plus.

B. L'erreur "hardware".

La défaillance d'un composant interne ou externe au système rendant le contenu de la mémoire centrale inutilisable, peut entraîner une action en "recovery". Il s'agit, par exemple :

- d'une coupure de courant,
- d'une défaillance du conditionnement d'air,
- d'une erreur de parité en mémoire centrale.

.....

D'autres erreurs, empêchant la suite normale de l'exploitation, sont possibles. Un cas particulier consiste en la destruction d'un ou de plusieurs fichiers critiques.

2.2.3.2. - Fonctionnement de la "recovery".

A. Le principe est de recommencer le traitement des transactions qui étaient actives au moment de l'interruption. Les transactions actives sont comprises dans l' "input-queue".

La restauration des transactions se base sur le fichier "beforelook". Elle comporte plusieurs étapes :

Première étape. - Recherche de la dernière "critical area"

La "critical area" représente la configuration du système à un instant donné. Cette zone, constituée en mémoire, est ajoutée au contenu de la FH-TAR au moment de chaque "clean-up" d'une transaction. Pour trouver la dernière "critical area", il suffit d'analyser, sur

tambour, les FH-TAR's des transactions qui sont passées par le "clean-up", c'est-à-dire celles qui sont en "out queue". Elle se trouvera logiquement dans la FH-TAR ayant le numéro de message le plus élevé.

Note.
En fait, la recherche s'effectuera sur les FH-TAR's mais également sur les FH-DBJ's qui sont l'équivalent des FH-TAR's pour les DBJ's et sur la RCV-AREA.
Si deux messages consécutifs proviennent du même terminal, il est nécessaire de sauver le contenu de la TAR ailleurs que dans la FH-TAR car celle-ci sera utilisée par le message "input" suivant. Cette zone de sauvetage constitue la RCV-AREA. Elle aussi peut contenir, à un instant donné, la dernière "critical area".

Deuxième_étape. - Restauration_des_"beforelooks".

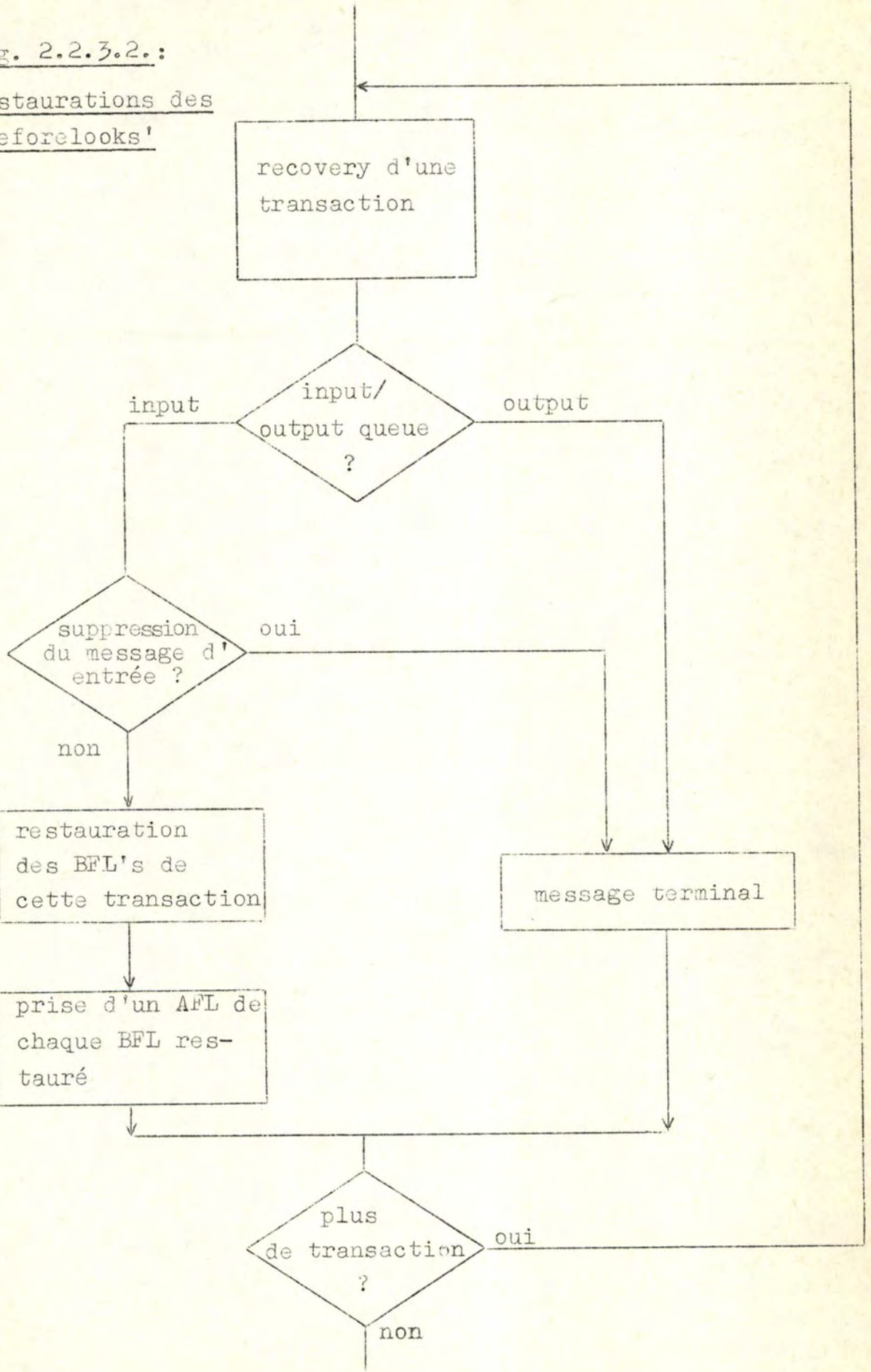
Le schéma de cette étape est décrit par la figure 2.2.3.2. L'"input queue" est composée de toutes les transactions en exécution ou en attente d'exécution. Ce sont ces transactions qu'il est nécessaire de relancer à leur point de départ.

Certaines précautions doivent être prises. En effet, certaines transactions traitées au moment de l'interruption, ont pu mettre à jour des enregistrements du fichier; il importe de replacer ces enregistrements dans l'état qu'ils représentaient au moment du dernier "clean-up". Il suffit, pour ce faire, de restaurer tous les "beforelooks" instaurés par ces transactions.

La mesure n'est pas suffisante. Pour que le fichier "afterlook" soit compatible, il suffira encore de supprimer les AFL's de ces transactions. Ce but sera réalisé en prenant un AFL de chaque BFL restauré.

En ce qui concerne les transactions se trouvant en "output queue", c'est-à-dire les transactions déjà traitées mais en attente de sortie des résultats, le cycle normal

fig. 2.2.3.2.:
Restaurations des 'beforelooks'



du traitement se poursuit. Un message spécial, indiquant la "recovery" est néanmoins lancé sur tous les terminaux de sortie.

Note.
La "recovery" des travaux "batch" ne se fait pas de façon automatique comme celle des transactions. Au moment du "restart", la console imprime les noms des travaux "batch" actifs au moment de l'interruption. Il appartiendra à l'opérateur de relancer ces travaux un à un.

Troisième étape. - Relance du cycle d'exécution.

Durant la "recovery", le "polling" a été interrompu. De même, certaines routines du RTS ont été suspendues. Il s'agit, maintenant, de réactiver ces routines (ex. : le "message communication handler") afin de rendre possible à nouveau l'envoi de messages. On réinitialise le cycle normal des transactions.

B. La procédure comporte, en fait, plusieurs options. Les plus fréquemment employées sont le "hot-restart" et le "cold-restart".

- "cold-restart" : utilisé lorsqu'il n'est plus possible de se baser sur l'état présent des fichiers critiques (fichiers FH-TAR, AFL, BFL, ...). Un "cold-restart" s'accompagne toujours d'une perte importante d'informations. Il est, de ce fait, une solution extrême. Il est également utilisé lors de l'initialisation du système.
- "hot-restart" : cette procédure correspond au mécanisme en trois étapes décrit ci-dessus. Il repose essentiellement sur les informations contenues dans les fichiers critiques. Le "hot-restart" n'engendre pas de perte d'informations ou très peu; il a seulement une influence défavorable sur le temps de réponse.

D'autres options secondaires existent :

- option "mod" : la "recovery" est précédée d'un certain nombre de modifications de programme.

- option "TPS" : le chargement préalable d'un ou de plusieurs programmes de traitement des messages est indispensable.

- option "AFL" : la destruction partielle ou totale d'une unité périphérique contenant les fichiers "application" exige une reconstruction immédiate. Celle-ci est une combinaison d'un rechargement du fichier à un certain point (dump) et d'une restauration de tous les mouvements ("afterlooks") encourus à partir de ce point.

2.2.3.3. - Problèmes de mise au point.

A. - Faut-il supprimer les messages de l'"input-queue" ?

Le schéma de la figure 2.2.3.2. fait apparaître un choix à l'opérateur. Ce dernier a la possibilité de supprimer tous les messages se trouvant dans l'"input-queue" au moment de l'interruption. L'erreur trouve son origine dans le programme de traitement d'un de ces messages. Généralement, la transaction fautive sera celle traitée en dernier lieu, juste avant l'interruption.

Le risque est grand,si aucune suppression de message n'est effectuée, de retomber dans l'erreur après la "recovery". Si l'on supprime tous les messages de l'"input-queue", certaines informations seront perdues. Par contre, la suppression du dernier message traité avant l'interruption semble fournir une solution idéale; le risque de retomber dans l'erreur y est faible et la perte d'informations est insignifiante.

B. - Quelle copie d'un fichier critique utilisera la "recovery" ?

Le "sysfile" se compose d'un certain nombre de fichiers critiques qui sont conservés en deux exemplaires sur deux unités "tambour" différentes. Lors de la "recovery", le problème se pose de savoir sur laquelle des deux copies d'un fichier critique s'effectuera l'analyse. Il en est ainsi pour la recherche de la "critical area", deux copies du fichier FH-TAR sont disponibles. De même, il y a deux copies des fichiers contenant les "beforelooks" et les "afterlooks".

Une copie peut être défectueuse ou non. Un mot d'état du fichier nous le renseignera. La solution apportée au problème par le Projet est basée sur un fichier "RCV" également conservé en deux exemplaires.

Si donc on désire utiliser un fichier critique, deux étapes sont nécessaires.

Première étape : choix de la copie de "RCV".

RCV contient tous les mots d'état des fichiers critiques. Il était nécessaire, en effet, de conserver ces informations ailleurs qu'en mémoire centrale car, en cas d'interruption, le contenu de cette mémoire est jugé inutilisable.

La figure 2.2.3.3.a. nous informe du contenu du fichier RCV.

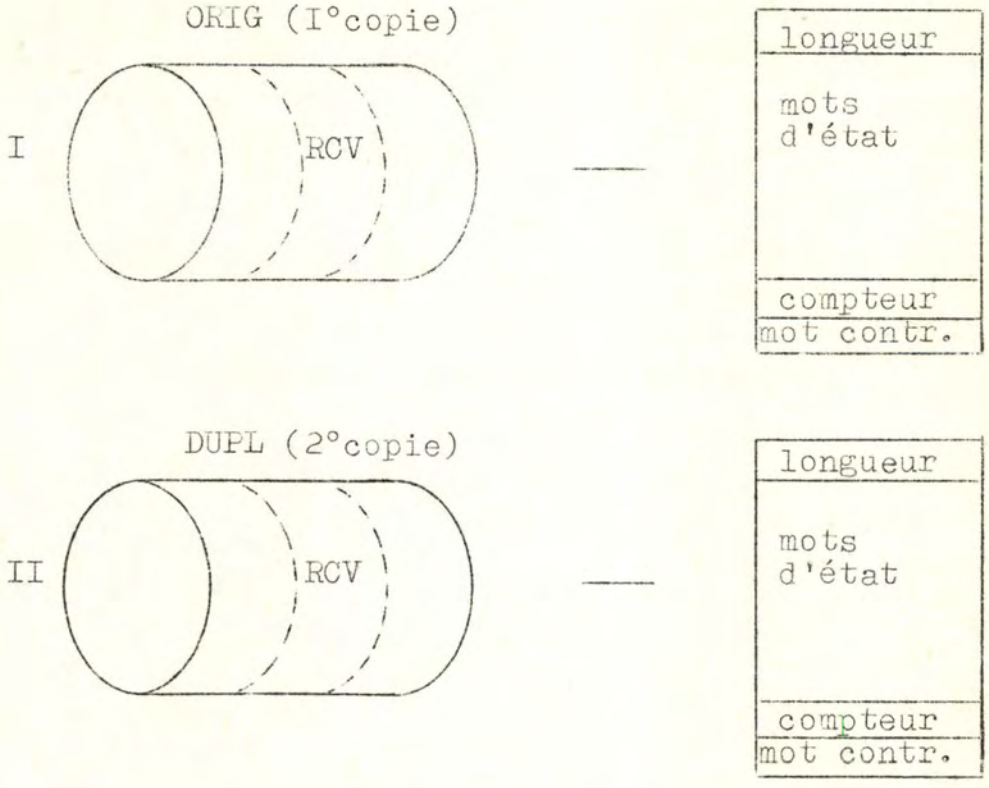


fig. 2.2.3.3.a. - fichier RCV.

- compteur : compteur d'utilisation de la table. Il est incrémenté de "1" à chaque utilisation.

- mot de contrôle : ce mot constitue la somme logique (bit par bit) de tous les mots d'état de la table.

Le choix de la copie du fichier RCV se fera de la façon décrite par la figure 2.2.3.3.b.

Les deux copies de RCV peuvent différer. En effet, la mise à jour des copies d'un fichier critique ne se fait pas simultanément; une interruption peut toujours intervenir entre la mise à jour des deux copies.

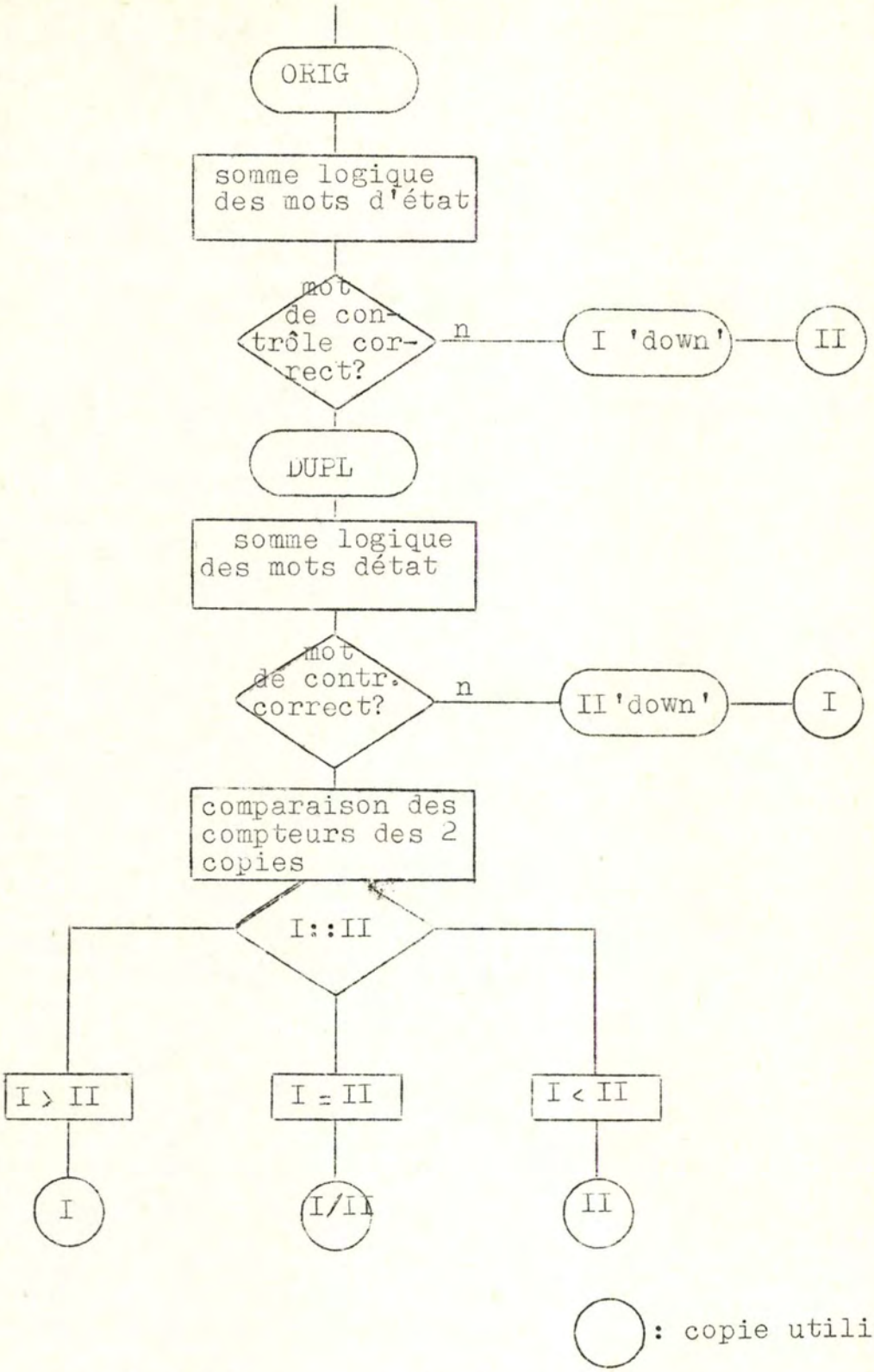


fig. 2.2.3.3.b. - organigramme de recherche de la copie RCV.

- les deux copies RCV contiennent leur propre mot d'état.
- une copie peut être "up" ou "down" suivant qu'elle est utilisable ou non. Cette information est contenue dans le mot d'état. Si le mot de contrôle d'une copie de RCV est incorrect, la copie concernée sera mise "down".
- si les deux copies de RCV ont leur mot de contrôle correct, il se peut qu'une des copies soit antérieure à l'autre. La comparaison des deux compteurs d'utilisation nous informera à ce sujet. Une copie peut être antérieure à l'autre si, au moment de l'interruption, on se situe entre la mise à jour des deux copies. Le choix de la copie dépendra, dès lors, de cette comparaison.

Deuxième_étape_:_choix_de_la_copie_du_fichier_critique.

Lorsque la copie du fichier RCV est choisie, un second choix est nécessaire quant à la copie du fichier critique que l'on désire utiliser. Il suffira, pour ce faire, d'examiner le mot de contrôle de ce fichier critique dans la copie du fichier RCV choisie en première étape. Le format d'un mot de contrôle permet de déceler immédiatement la copie à utiliser :

type	Ière copie	2me copie
A	B	C

- le type du fichier (A) indique si celui-ci est conservé en une ou deux copies.
- les zones B et C nous informent sur l'état "up" ou "down" des deux copies.
- si le fichier ne possède qu'une seule copie, la zone C est vide.

C. - Comment effectuer la restauration d'une copie "down" d'un fichier critique ?

Cette situation nous ramène à une idée déjà exprimée et qui concerne le choix qui se présente à l'utilisateur quant à la continuité de l'exploitation. Le Projet a opté pour la sécurité. C'est pourquoi, en cas de défectuosité d'une copie d'un fichier critique, le système s'interrompt afin de permettre la restauration immédiate de la copie défaillante.

La procédure est décrite par la figure 2.2.3.3.c. Elle consiste à écrire la totalité de "sysfile" sur bande (dump "panic") à partir de la copie correcte et de réécrire le fichier critique (sous-fichier de "sysfile") à partir de cette bande à l'endroit correct sur le tambour contenant la copie défectueuse. Si cette écriture échoue, la totalité de "sysfile" sera réécrite à un endroit différent sur ce tambour.

D. - Imperfections du système.

1. Toute "recovery" comporte une phase de repositionnement des bandes "logging". Il s'agit, en effet, de replacer les bandes de l'exploitation à un endroit correct afin de rendre possible la suite des opérations. Or, cette opération exige que soit relu préalablement le label de chacune de ces bandes afin de s'assurer que celles-ci correspondent effectivement à celles que l'on désire utiliser. Une telle opération et le repositionnement qui s'ensuit prend généralement 7 à 8 minutes, ce qui, en cas de "recovery", constitue une consommation de temps considérable.

   Une solution au problème consisterait à introduire, dans la bande et à intervalles réguliers, un enregistrement particulier reprenant certaines informations contenues dans le label; le gain de temps réalisé compenserait aisément la place nécessaire à ces enregistrements.

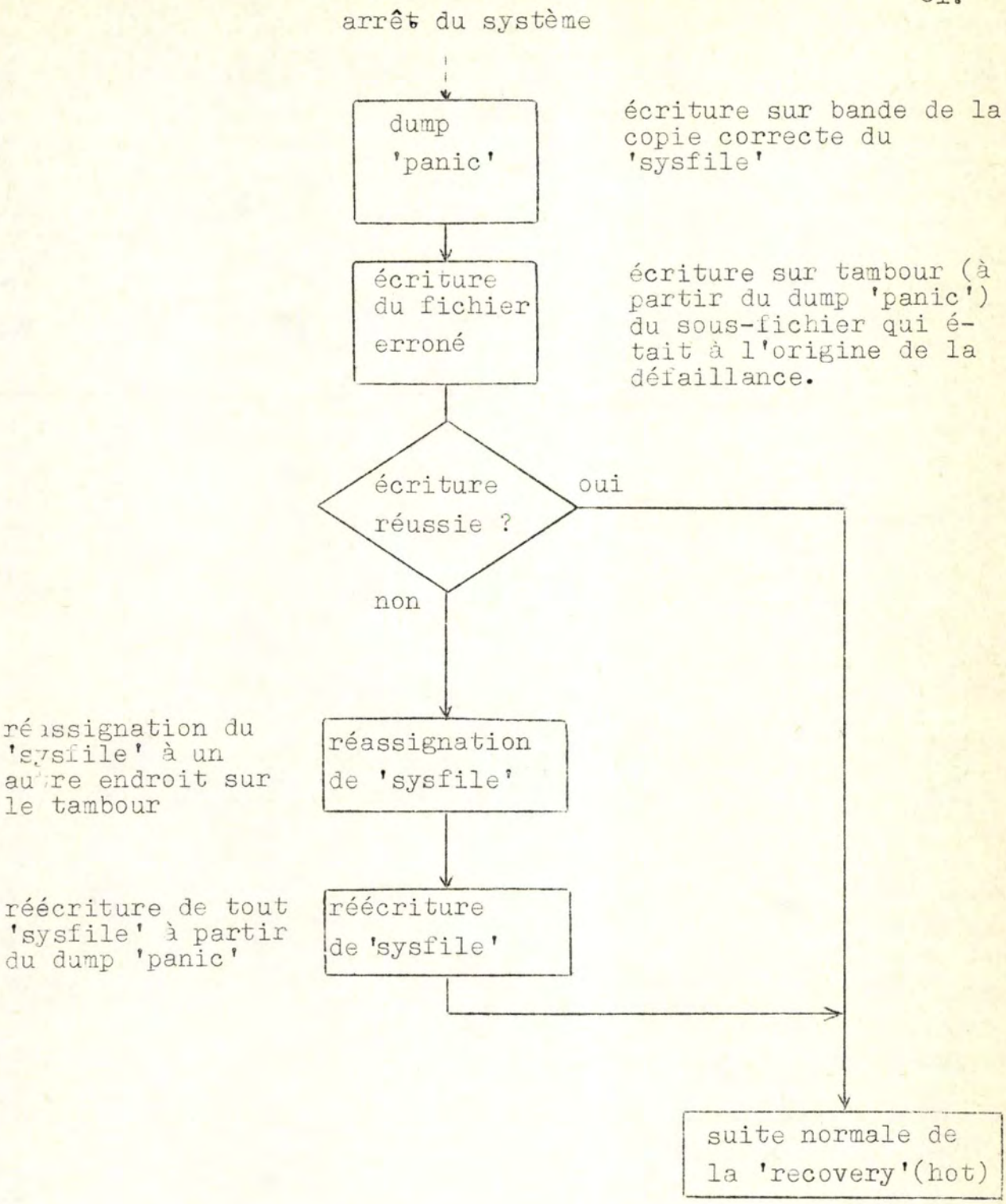


fig. 2.2.3.3.c. Recovery d'une copie d'un fichier critique

2. Le "cold restart" fait souvent l'objet d'une utilisation abusive. Il est une solution de facilité où le volume des informations perdues est considérable.

   Il doit être utilisé dans les cas extrêmes où tout autre moyen s'est avéré infructueux. A la limite, son utilisation devrait être interdite.

2.2.4. - Quelques mesures supplémentaires.

. Le RTOS possède, dans sa librairie, certaines routines dont la suppression serait souhaitable, notamment :

  - la routine "INC" : cette routine a pour but d'effectuer des modifications dynamiques des zones de mémoire centrale.
  - la routine "DIN" : comme "INC", cette routine exécute des modifications dynamiques mais celles-ci se font sur tambour.

. Les messages "console" concernant des erreurs très graves seront rendus très spectaculaires afin d'attirer plus encore l'attention de l'opérateur (double encadrement sur le listing de sortie, sonnerie d'alarme, ..)

. Interdire le chargement d'un fichier pendant l'exploitation du temps-réel, surtout s'il s'agit d'un fichier "critique".
  Exemple : la mauvaise lecture d'une carte de contrôle, lors du chargement d'une partie de "sysfile" alors que ce dernier était actif, a engendré la destruction des "beforelooks", "TAR's", ...

- Respect des priorités. Il faut, à tout prix, protéger la priorité du RTS afin de garder au traitement des transaction, la prédominance qu'il exige. Il n'est pas question qu'un programme, en "batch pur", ne s'exécute pas sous la même priorité que le RTS.

- Il est utile de prendre une copie du fichier des TPS's chaque fois que celui-ci fait l'objet de modifications ou d'un nouvel assemblage.

- Certaines transactions non terminées au "choc down" doivent pouvoir s'achever à la reprise de l'exploitation le jour suivant. Afin de permettre cette éventualité, le "SYSFILE" est vidé sur bande en soirée et restauré tel quel en début d'exploitation.

.....

Section 3. - Mesures "organisationnelles".

Il sera question, dans cette section, d'un certain nombre de problèmes qui relèvent plus d'une attitude d'esprit que d'une description de différents moyens d'exécution, comme cela a été le cas dans les sections précédentes. Nous envisagerons l'aspect "sécurité" dans le cadre de l'organisation interne d'un centre de traitement de l'information.

Sans qu'il soit question de remettre en cause les finalités mêmes du système, nous tenterons d'analyser certains aspects critiques touchant notamment à certains choix de base du système. Il sera question des problèmes suivants :

- concurrence du "temps-réel" et des travaux "batch"
- protection du secret des données
- mise au point du Système.

Une partie sera consacrée à des éléments d'organisation de l'exploitation. Il s'agira tout autant de considérer des attitudes humaines que des dispositions pratiques. Une bonne organisation est le point de départ d'un système sûr et fiable. La mise en oeuvre des éléments analysés dans les sections précédentes serait tout à fait inefficace si ces éléments ne se pliaient pas à certaines obligations particulières d'organisation interne.

2.3.1. - Activités concurrentes : le temps-réel et les travaux "batch".

La vulnérabilité des fichiers "application" provient, dans une large mesure, de la possibilité qu'ont les programmes "batch" de pouvoir modifier le contenu de ces

fichiers à l'insu du temps-réel lui-même, c'est-à-dire à l'insu du RTS.

Le RTS utilise, pour son exploitation, le procédé de la "lock list" afin de gérer les conflits d'accès aux enregistrements.

Les programmes s'exécutant en-dehors du RTS, ignorent la gestion interne de celui-ci et, de ce fait, le mécanisme de la "lock list".

Cette situation peut amener une incertitude quant au contenu des enregistrements :

- Si un programme "batch" modifiant le contenu d'un enregistrement est interrompu par une transaction du temps-réel (plus prioritaire) dont l'objectif est de mettre à jour le même enregistrement, l'état final de l'enregistrement sera incertain.

- Les programmes en différé (sous RTOS) n'utilisent pas les "file handlers". Il appartient au programmeur lui-même d'effectuer tous les contrôles, ce qui augmente le risque d'erreurs humaines.

- Tout comme la "lock list", la "recovery" est interne au RTS. Mais si le contenu des enregistrements devient incertain, toute "recovery" basée sur la restauration des "beforelooks" et "afterlooks" devient inefficace.

Afin de résoudre le problème, un troisième type de programme fut créé : il s'agit des DBJ's (Data Base Jobs).

Un DBJ constitue un "run" spécial et figure dans le répertoire des programmes du RTS. (1)

Les DBJ's disposent de toutes les facilités du RTS et, notamment, de la "lock list". Ils sont également repris dans le système de "recovery" (voir les "procédures de secours").

Tout comme les transactions du temps-réel, les DBJ's ont la possibilité de modifier les fichiers "application" mais ils disposent, à l'intérieur du RTS, d'une priorité inférieure à celle des transactions. L'exécution d'un programme DBJ pourra donc être interrompue par le temps-réel (les transactions). Mais si un tel programme exécute une modification sur un enregistrement du fichier, la transaction qui l'interrompt ne pourra effectuer de mise à jour sur cet enregistrement et elle sera placée en attente (principe du "file lock").

Les DBJ's ne résolvent pas tout. Il est impensable de supprimer, purement et simplement, le "batch pur". En effet, certains travaux, vu leur type et leur période d'exécution, doivent s'exécuter en "batch pur". Exemples :

- programmes de tris
- assemblages

Devant cette situation, d'autres mesures sont, dès lors, nécessaires. Deux possibilités :

---

(1) L'ordre de chargement d'un DBJ est soumis au RTOS qui a le droit d'interdire le chargement s'il juge que la place en mémoire centrale est insuffisante. Avant de recevoir, pour la Ière fois, le contrôle, le DBJ est considéré comme un programme de "batch pur" avec la priorité la plus basse. Au moment où, pour la première fois, il prend le contrôle, il se déclare au RTS et il reçoit la même priorité que ce dernier (classe 1 mid.1)

A) interdire, purement et simplement, l'accès aux fichiers "application" par les travaux en "batch pur".

Cette interdiction est possible en modifiant la routine d'assignation du RTOS. Tout programme en "batch pur" se verra dans l'impossibilité de s'assigner un fichier "application". Cette modification du RTOS est réalisable; elle serait, d'ailleurs, la seule modification "durable" du RTOS.

L'interdiction peut se faire également en testant un "indicateur" associé à chaque fichier dans le catalogue des fichiers. Un bit "on" signifierait que le fichier ne pourra être assigné par un programme en "batch pur".

Remarque : il importe qu'aucun des programmes en "batch pur" ne se donne le nom de "RTS".

B) permettre l'accès aux fichiers "application" par n'importe quel type de programme.

Cette possibilité peut être acceptée si un système de "lock list" se situe au niveau du RTOS. La modification du RTOS constituerait une opération délicate et dangereuse.

La solution ne résout cependant pas le problème de la "recovery". Rappelons, en effet, qu'une modification d'un enregistrement d'un fichier "application" par un programme en "batch pur" s'exécute sans prendre de "beforelook" et d' "afterlook". Toute "recovery", dès lors, perd toute son efficacité.

2.3.2. - Protection du secret des données.

Le contrôle de l'accès à l'information ne se pose pas encore de façon urgente à l'A.N.M.C. Actuellement,

l'information est essentiellement administrative et comptable. La seule protection existante est le secret financier de certaines informations.

Les différentes organisations (fédération ou hôpital) n'ont accès qu'aux informations concernant leur administration. Cette protection est réalisée au moyen d'un contrôle sur l'identité des terminaux. Ce contrôle est purement "hardware" et gère l'accès à un groupe d'informations donné dans le fichier.

Certaines exceptions à la règle sont prévues. Le passage d'un individu d'une fédération à une autre s'effectuera au moyen d'une transaction spéciale ayant le droit d'accéder aux données de plusieurs fédérations.

Dans le cas où le Projet envisagerait de traiter des informations strictement confidentielles (ex. diagnostics de médecins), la question se pose à l'analyste de savoir "qui est qualifié pour ajouter, modifier ou supprimer des enregistrements ?".

Notre intention n'est pas de décrire un modèle type de protection de la CONFIDENTIALITE des données. D'excellents ouvrages ont abondamment et fort bien traité cette matière, tels que :

-"Security, accuracy and privacy in computer systems" de James Martin (chap. sur les schémas d'autorisation).

-"On the implementation of security measures in information systems". (Communication of the ACM) - Revue du mois d'avril 1972, volume 15, n° 4).

-"Vertrouwelijkheid medische gegevens in verband met databanken" de J.N.HERBSHLEB.

Il sera question, dans ces pages, d'introduire certains préalables sémantiques indispensables à la mise en oeuvre d'un tel système de protection.

Ces considérations constituent les bases d'une approche plus technique du problème. Certaines orientations du point de vue de l'implémentation seront avancées.

2.3.2.1. - Préalables sémantiques.

La confidentialité des données, dans le système de l'A.N.M.C., concerne des données de types médical, comptable et administratif. Il sera donc, avant tout, question de définir quelles sont les informations relevant de ces trois groupes.

Une deuxième étape consistera à résoudre le problème des "qualifications" : qui est qualifié à consulter, ajouter, mettre à jour ou supprimer les données ?

Une dernière étape prendra en considération certains problèmes secondaires découlant du caractère spécial des données.

Première étape : les données.

S'il est assez aisé de définir si une donnée est de type administratif ou comptable, il en va autrement pour les données de type médical. La frontière entre ces différents groupes de données est parfois difficile à établir.

Il est clair que certaines données apparaissent immédiatement comme non-médicales, comme la nature de l'assurance d'un patient, le règlement de sa note de frais d'hospitalisation. Par contre, beaucoup de données qui, à première vue, apparaissent de nature administrative, peuvent néanmoins avoir une importance du point de vue médical.

Exemples :- dates de début et de fin d'hospitalisation.
- adresse d'un individu qui n'est plus hospitalisé mais qui a encore besoin de soins.

Les données médicales ne constituent donc pas un groupe homogène. Le système de protection doit être en mesure de distinguer la nature des données.

Deuxième étape : les qualifications.

Personne n'est qualifié à consulter toutes les données d'une banque de données. En général, on est habilité à utiliser des données dans la mesure où la fonction que l'on exécute l'exige.

Il n'est pas seulement question de déterminer qui a le droit de consulter les données; il faut également décider par qui ces données peuvent être créées, modifiées ou supprimées.

AJOUTER :

- le personnel "médical" peut introduire des données accessibles à d'autres personnes et des données strictement privées que seul, il se réserve le droit de consulter.
- lors de la création (l'introduction des données), il sera spécifié les personnes ou les catégories de personnes qui pourront consulter ces données.
- les avis des infirmiers(infirmières) pourraient être traités au même titre que les données du médecin pour lequel ils(elles) exercent leur fonction.
- lors de la création des données, il sera également défini les données strictement privées et les données qui, dans certains cas très urgents (ex. décès ou retrait d'un médecin) pourraient être consultées par d'autres personnes.

CONSULTATION :

- les données d'une fédération ou d'un hôpital ne seront accessibles qu'aux membres respectifs de ces institutions.

. la consultation est dépendante de la fonction exercée par le demandeur.

. il sera demandé l'identité du demandeur, dans le cas de consultation de données qu'il n'a pas introduites, afin de déterminer si sa demande pourra être satisfaite.

MODIFICATION :

. l'administration peut modifier des données administratives et médico-administratives.

. les diagnostics des médecins ne peuvent être changés.

. il sera egalement déterminé si, au décès d'un médecin, un suppléant sera désigné ou si les données resteront inaccessibles.

SUPPRIMER :

. seules, les données médicales exigent, une fois de plus, une attention particulière.

. en cas de décès d'un patient, les données concernant ce patient exigent une protection au même titre que celles des patients en traitement.

. afin d'éviter toute fraude, les nom, adresse, etc... seront supprimés du fichier en cas de décès du patient; les médecins concernés indiqueront quelles sont les données dont la conservation s'impose et quelle sont celles auxquelles on pourra accéder ou non.

Troisième étape : les problèmes annexes.

En aucun cas, une correspondance entre une maladie déterminée et des noms de patients ne pourra être perçue.

De même, certaines données administratives (traitements, impôts, ...) ne pourront être visibles sous aucune forme.

2.3.2.2. - Schémas de solutions.

Suivant le type de l'application étudiée, la notion d'utilisateur peut évoluer et se rapporter à :

- un individu
- une catégorie d'individus
- un programme d'application (TPS's)
- un terminal
- une combinaison des cas précédents.

De même, plutôt que de baser la décision sur l'accès ou le non-accès des données, il peut être préférable, dans certains cas, de structurer les règles de décision autour de certaines facilités du système, telles que :

- les programmes (TPS's)
- les types de transactions
- les fichiers
- les volumes (disque, tambours, ...)

James MARTIN, dans "Security, accuracy and privacy" préconise quatre méthodes fondamentales :

1. stratification : division horizontale de l'information en niveaux tels que "Top Secret", "Secret", "Confidentiel", ...
2. compartimentalisation : l'information est divisée verticalement entre les utilisateurs. Il s'agit d'une partition de l'information de telle sorte que les données de l'utilisateur A ne sont accessibles à aucun autre utilisateur.
3. table d'autorisation : le système de protection repose sur une table qui indique ce que chaque utilisateur est habilité à faire.
4. information de protection dans les enregistrements : ces informations indiquent qui a la possibilité de lire, de modifier ou de supprimer un enregistrement.

Ce contrôle peut se placer dans les enregistrements eux-mêmes ou dans l'index utilisé pour adresser les enregistrements.

Note : l'emploi des mots de passe ne donne pas un haut niveau de sécurité; cependant, la combinaison de ce procédé avec d'autres précautions peut être extrêmement fiable.

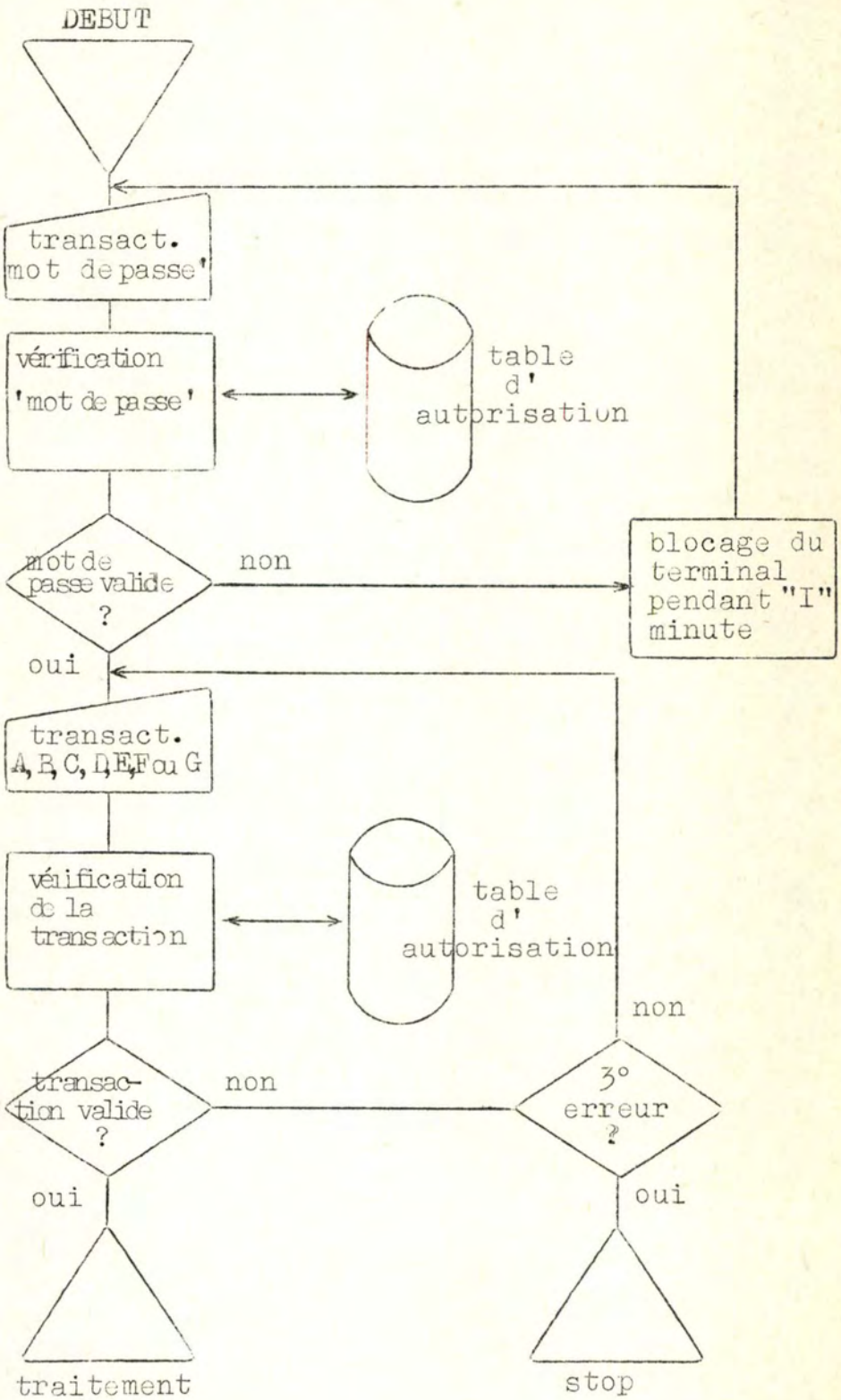
.....

Souvent, les solutions adoptées constituent une combinaison de ces méthodes de base. On pourrait ainsi définir un système de protection en se basant sur les deux dernières méthodes. La "matrice de sécurité" ou "table d'autorisation" aurait le format suivant :

	N° Ident.	Mot de passe	types de TRANSACTIONS						
			A	B	C	D	E	F	G
utilisateur I					x	x			
" II			x		x		x		
" III				x					
" IV				x	x	x			
" V									x
" VI					x		x	x	
" VII			x						

- les "colonnes" définissent les différents types de transactions possibles.
- les "lignes" établissent les utilisateurs du système.
- chaque utilisateur possède son mot de passe, de même qu'un numéro unique d'identification attribué par le système.
- chaque type de transaction permis à un utilisateur constitue un élément non vide (x) de la matrice d'autorisation.

. l'organigramme d'exploitation associé à cette matrice pourrait être le suivant :



. Il est possible de rendre l'organigramme plus complexe encore. D'autre part, le contenu de la matrice peut être plus élaboré. On pourrait ainsi imaginer que les éléments de la matrice comportent des indications quant aux possibilités de consultation, modification et suppression qu'auront les utilisateurs dans les différents types de transactions.

	N° Ident.	Mot de passe	types de TRANSACTIONS						
			A	B	C	D	E	F	G
utilisateurI					c,m	c,m			
" II			c		c		c,m,s		
" III				c,m					
" IV				c,m,s	c	c			
" V									c
" VI					c,m		c	c,m	
" VII			c,m,s						

exemples : .l'utilisateur IV aura la possibilité de faire de la consultation s'il emploie la transaction D.

.par contre, l'utilisateur I, pour le même type de transaction (D), pourra à la fois consulter et modifier.

. S'il fallait appliquer cette solution à l'A.N.M.C., d'autres mesures supplémentaires seraient indispensables.

A titre d'exemples :

1) pour éviter qu'un médecin ne consulte les données relatives à un patient qu'il n'a pas traité, une information supplémentaire (un numéro d'identification ou le mot de passe) serait ajoutée, soit dans l'enregistrement du patient, soit au niveau de l'index ("bucket").

2) désignation d'un responsable qui, seul, aurait la possibilité de modifier la table d'autorisation afin de résoudre les problèmes de retrait ou de décès d'un médecin, l'ajoute de nouvelles transactions, ...

. Le choix d'une solution doit se baser sur un juste partage qui doit s'opérer entre les risques de fraude et le coût des opérations de sécurité.

.....

Signalons que ce point 2.3.2 ne prétend nullement apporter une solution définitive concernant la protection du secret des données. Il s'agit, avant tout, d'un ensemble de suggestions destinées à soulever la réflexion et à engager la discussion parmi les membres du Projet confrontés au problème.

.....

2.3.3. - Mise au point du système.

Une des préoccupations de la maintenance s'attache à la gestion et à la tenue à jour des programmes. Voyons comment est régie la modification d'un programme :

2.3.3.1. - Règles de modification.

Les modifications se font par l'intermédiaire d'un utilitaire auquel on fournit des cartes "MOD". Celles-ci ont pour but de modifier un programme directement sur tambour et en code "objet". Cette correction, en mode absolu, se fait dès lors automatiquement sur les quatre versions de la page.

Afin d'éviter les mises à jour illicites, le programmeur se voit obligé d'introduire en premier lieu la valeur

ancienne de la position à modifier (adresse virtuelle). Si cette valeur ne correspond pas à celle détectée par l'ordinateur à la même adresse, un message d'erreur est imprimé et l'exécution n'est pas effectuée.

Si la modification est mineure, elle se fera en temps-réel.

Plusieurs modifications d'un même programme peuvent amener le réassemblage du programme.

2.3.3.2. - Inconvénients de la règle.

L'opérateur à la console a la possibilité de se soustraire à la règle d'introduction préalable de la valeur ancienne. Il s'agit, comme pour les "labels", de cette option particulière qui permet à certains contrôles, même importants, d'être contournés de sorte qu'ils perdent toute leur efficacité.

La modification, en mode absolu, d'une page peut coïncider avec l'exécution en mémoire centrale, de la même page. La transaction, active à ce moment, ne profitera pas de l'amélioration qui est apportée à la page virtuelle.

2.3.3.3. - Considérations.

S'il est difficile d'interdire que des modifications soient effectuées, on peut néanmoins exiger que le programmeur passe plus de temps à tester son programme. En effet, s'il a la possibilité de corriger ultérieurement son programme, cela peut entraîner, chez lui, une certaine négligence durant les tests.

Il peut être souhaitable, dès lors, que les modifications soient exécutées par une seule et même personne (ex. : par le chef du groupe de programmation). Cette personne pourra juger ainsi les qualifications de son

personnel et déceler ses faiblesses éventuelles (ex. : une répétition anormale d'erreurs).

L'ancienne copie d'un programme ayant fait l'objet d'une modification (peut-être d'un réassemblage) sera conservée. Un retour à la situation ancienne peut toujours être exigé d'une façon ou d'une autre.

D'autres précautions sont utiles, notamment le catalogage du nouvel objet et la suppression de l'ancien dans la librairie du système.

Les modifications de programmes, de même que les chargements de nouveaux programmes s'effectueront en-dehors du "temps-réel" et cela, de façon à éviter toute incertitude quant au contenu de la copie utilisée lors de l'exécution d'une demande pour de tels programmes. (exemple : cas d'un programme exécuté en mémoire centrale simultanément à sa modification sur tambour).

Ces actions peuvent paraître dérisoires en apparence mais insistons sur le fait qu'une négligence, dans ce domaine, peut avoir des conséquences disproportionnées par rapport à son origine.

A titre d'exemple : le mauvais catalogage du nouvel "objet" peut entraîner des conséquences désastreuses. Il est fondamental que le temps et le soin que requièrent ces tâches soient pris sérieusement en considération.

Notes sur les tests.

Le système se doit d'être protégé contre les violations que pourraient provoquer les tests. Le Projet procure, à cet effet, une protection suffisante :

- un simulateur du "temps-réel" est utilisé, ce qui permet d'exécuter les tests sans qu'il soit possible de modifier les fichiers "application".

- il est possible, également, de travailler en mode "test". Ce mode permet d'utiliser les programmes du "temps-réel" sans danger de modification des fichiers. Certains tests s'appuient sur des fichiers "tests" spécialement conçus à cet effet.

2.3.4. - Eléments de l'organisation d'exploitation.

L'exploitation d'un système rentable et efficace exige que cette exploitation s'appuie sur une organisation impeccable. Certains éléments de cette organisation sont d'autant plus indispensables que le système travaille en "temps-réel". A titre d'exemples, le planning des travaux en différé et le rôle d'un bibliothécaire des bandes sont deux éléments qui revêtent une importance considérable dans l'organisation de la production d'un système en "temps-réel". L'existence d'un système sûr d'information implique donc la présence d'une organisation bien structurée qui garantisse la bonne exploitation du centre.

Les structures d'organisation font partie intégrante de la mise en oeuvre du système. L'analyse qui va suivre portera sur certains éléments d'organisation, spécifiques à tout système en "temps-réel" et qu'il nous a semblé utile de rappeler dans le cadre de l'ANMC. Ces éléments concernent essentiellement l'organisation de la production.

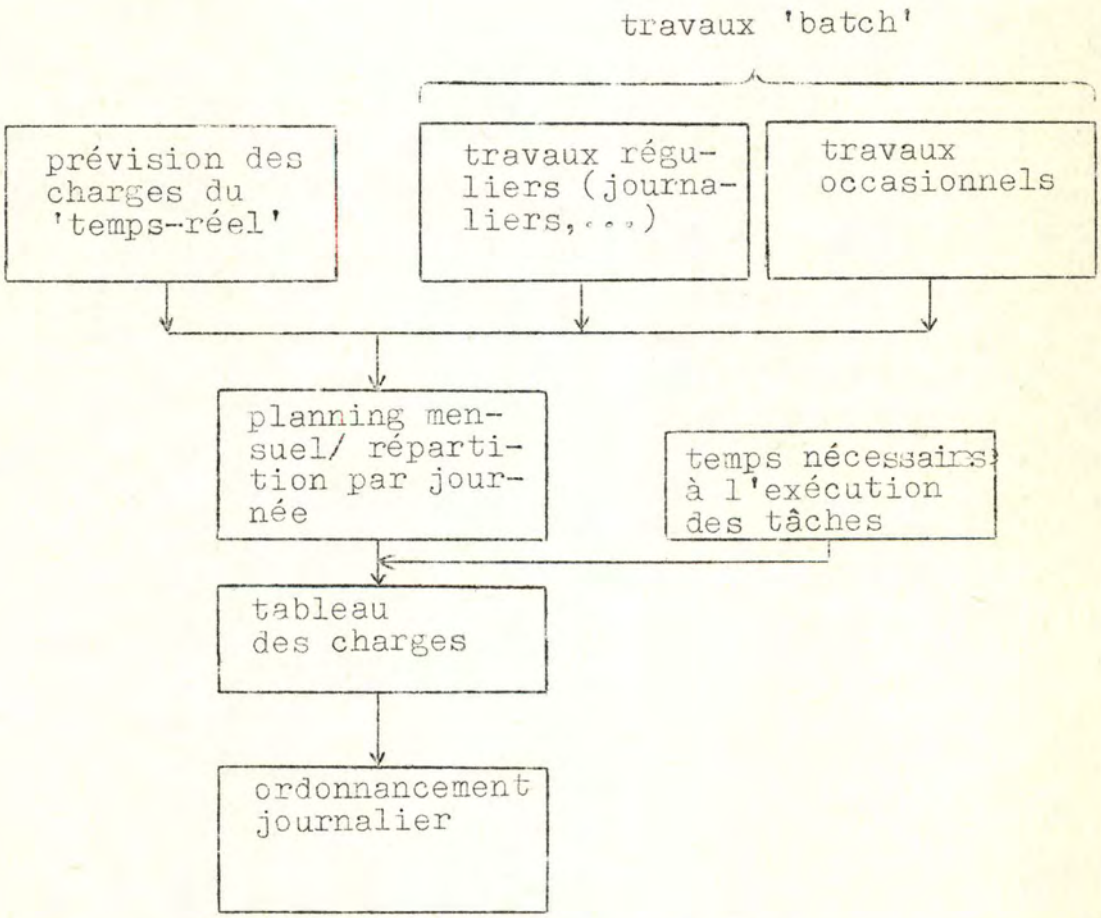
2.3.4.1. - La planification des travaux en différé.

Le temps-réel se caractérise par la coexistence de travaux interactifs et en différé. La planification aura pour but l'enchaînement harmonieux et continu des différents travaux en fonction des ressources offertes par le système (CPU, taille mémoire, périphériques, ...) et des besoins de ces travaux.

La planification des travaux "batch" consiste à gérer une série de jobs qui se répartissent en plusieurs catégories selon leur rythme (journalier, hebdomadaire, mensuel, etc...) et selon qu'ils sont réguliers ou non. Le but final est d'arriver à établir un planning mensuel avec la répartition, par journée, des tâches et en tenant compte des exigences du temps-réel.

On en déduit un tableau des charges en prenant en considération les temps nécessaires à l'exécution des différentes tâches et des besoins en place mémoire de ces tâches.

Ce tableau des charges permet de réaliser un ordonnancement du jour avec, comme objectif principal, la meilleure utilisation des ressources du système. Cet ordonnancement s'accompagnera d'une gestion des unités périphériques.



ordonnancement des travaux "batch"

L'ordonnancement journalier final se présentera sous la forme d'un tableau dont chaque colonne représenterait une unité disque ou bande, la colonne de gauche étant réservée à l'énumération des différents programmes (ou succession d'opérations).

Cet ordonnancement est indispensable à tout système évoluant en multiprogrammation.

2.3.4.2. - Gestion de la bibliothèque.

Un gestionnaire de la bibliothèque des volumes bandes et disques s'inspirera du tableau des charges afin de s'accorder avec la marche des opérations.

Le travail du gestionnaire est essentiel. La sécurité des opérations et la protection des informations dépendront du soin qu'il apportera à ses activités que l'on peut classer selon deux types :

1. des travaux de manutention : notamment, la mise à jour manuelle, à tout moment de la journée, du fichier des bandes.
2. des travaux de préparation : il s'agit d'un travail de sélection des supports intervenant dans les différents travaux (temps-réel et batch)

Il n'est pas utile de s'étendre davantage sur ce problème dans le cadre de cette étude; insistons néanmoins sur le fait qu'il constitue un élément prédominant du point de vue de la sécurité et qu'il doit être considéré dans sa juste mesure.

Note. - Il existe, depuis peu, des procédures de gestion des bibliothèques de bandes par ordinateur, excluant, de ce fait, de nombreux risques d'erreurs (humaines, en particulier). Les buts poursuivis par ces procédures sont :

- la suppression de l'étiquetage des bobines par les opérateurs (source d'erreurs et de pertes de temps);
- la suppression du fichier mis à jour manuellement à tout moment de la journée qui, pour des bibliothèques importantes, nécessite souvent une ou plusieurs personnes à temps complet sans pour cela donner une vue d'ensemble rapide et nette de la situation de la bibliothèque, à un instant donné.

2.3.4.3. - La programmation de base.

La programmation doit pouvoir offrir un large éventail de possibilités de test. Il est parfois utile de perdre du temps à la programmation de façon à pouvoir détecter plus rapidement les erreurs. Si ces vérifications automatiques paraissent utiles, elles se traduisent par un alourdissement des programmes. Il faut donc pouvoir discerner un équilibre entre la charge que représentent ces contrôles et le risque encourusi ceux-ci n'existaient pas.

Il sera seulement question, dans ce point, de quelques souhaits fort utiles :

1. un contrôle plus serré de toutes les opérations d'entrées et sorties et, surtout, les ordres d'écritures.
2. vérification de tous les domaines d'un code. A titre d'exemple : si A, B et C sont des codes permis, il conviendra de tester successivement les trois codes, de même la possibilité que le code soit erroné c'est-à-dire différent de A, B ou C.
3. visualisation de la comptabilité de certains totaux concernant soit le temps-réel (total des transactions traitées depuis le début de la journée, total des

messages entrés par lignes, ...) soit des travaux "batch" (exemple dans un programme de facturation : visualisation de totaux qui seront comparés avec un contrôle manuel simultané).

4. qualité et présentation des instructions - importance des commentaires.

5. standardisation de la terminologie. Exemple rencontré à l'ANMC : deux routines contrôlant respectivement les labels de bandes des travaux "batch" (DBJ's) et du temps-réel exigent des réponses différentes de la part de l'opérateur à la console.

6. quelques outils d'analyse utiles au travail du programmeur.

DUMP

Il s'agit d'un programme de restitution complète de la mémoire.

TRACE

Le "TRACE" est un programme de "suivi" dynamique (sorte de "pas à pas" de l'exécution).
Il y a plusieurs types de "TRACE". Le plus utilisé est un programme de localisation d'instructions qui intervient à chaque instruction exécutée. Le programme imprime les contenus des différents registres de base du système après l'exécution d'une instruction.

BIT-MAP

Dans certains cas, le programmeur se crée lui-même des points de repère. Ce sont des indicateurs que l'on incrémente lors du passage de l'exécution en un point du programme. Ces repères portent le nom de "bit-map" dans le projet RTS.

Ces trois types de programme de service sont utilisés à l'ANMC.

7. direction de la programmation. - Les problèmes que pose la direction de la programmation sont beaucoup plus complexes dans un système en temps-réel que dans d'autres types d'installation. La discipline doit être plus rigoureuse. Dans un système en temps-réel, on travaille sur plusieurs programmes à la fois. Ces programmes sont des petites parties qui constitueront un ensemble étroitement intégré. Les segments de programmes écrits séparément devront, à un moment donné, s'emboîter les uns dans les autres.

Le directeur de la programmation aura, comme objectif primordial, de coordonner les interactions complexes entre les travaux des différents analystes-programmeurs.

Son travail aura deux aspects :

A. Il devra s'assurer que les travaux de ses différents programmeurs ne s'écartent pas des spécifications initiales concernant la conception originale du système.

Le programmeur réalise son programme en prenant un certain nombre de décisions qui pourraient s'écarter du schéma initial.

Exemples de contraintes :

- légèreté des programmes (contrainte du temps de réponse)
- respect des priorités
- fiabilité élevée
- contrainte de place mémoire
- .....

B. L'agencement des différents programmes constitue le second aspect. Si un programmeur effectue une modification dans son programme, elle va vraisemblablement affecter le travail d'autres programmeurs. La communication entre les différents membres de l'équipe se fera par le biais du chef de la programmation.

2.3.4.4. - Réflexions.

A - Travail de groupe.

Un groupe "software" est généralement composé de jeunes analystes-programmeurs où chacun doit accepter le concept du travail en équipe et percevoir clairement son rôle face à ceux des autres membres de l'équipe.

Tous les programmeurs ne s'adaptent pas nécessairement au travail en équipe. Un très bon programmeur peut ne pas convenir dans une équipe "temps-réel".

Soulignons, par ailleurs, l'importance d'une éducation qui soit constituée à la fois d'une formation spécialisée et d'une formation générale. Si la première se conçoit logiquement, la seconde n'en est pas moins aussi indispensable.

L'importance du chef d'équipe est, à nouveau, mise en évidence. Il est attaché à la solution des problèmes et communique son enthousiasme à son équipe. C'est un homme qui doit cependant faire confiance à tous ceux qui l'entourent. Il importe, d'ailleurs, que le sentiment de confiance soit partagé par tous les membres de l'équipe.

B - Séparation des tâches.

Un aspect important de la bonne exploitation d'un centre informatique est la "séparation des tâches".

Si une communauté de vue doit exister entre les différentes sections (analystes, programmeur, gestionnaire de la bibliothèque, opérateurs, ...), il importe néanmoins que chacune d'elles garde ses caractéristiques propres.

==========

C H A P I T R E III.

SAUVETAGE D'UN FICHIER VOLUMINEUX.

Lorsqu'un fichier devient trop volumineux, il n'est plus possible, pour des raisons pécuniaires, de le conserver en deux copies. De même un dump journalier de l'ensemble du fichier ne peut plus s'effectuer, vu le temps que nécessiterait son exécution.

La présente analyse tentera d'apporter des solutions au problème. L'éventualité d'une exploitation "24 heures sur 24" sera également prise en considération à la fin de cette étude.

Section 1. - Données techniques.

3.1.1. - Les bandes.

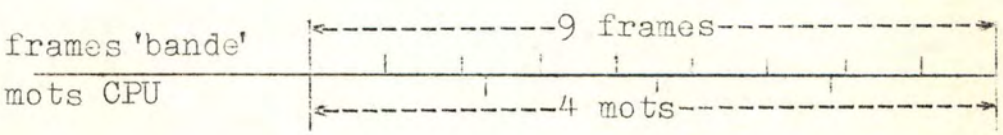
Le type de bande utilisé est l'UNISERVO 12.

Caractéristiques.

longueur de la bande	730 mètres
densité	1600 frames/inch
nombre de "tracks"	9 (8 données, 1 parité)
espace inter-bloc	0,6 inch ou 1,5 cm.
nombre d'unités par sous-système	2 à 16

Format des données.

Le transfert des données vers ou en provenance du périphérique s'effectue par multiples de 4 mots de 18 bits.



De cette manière . 4 mots = 9 FRAMES
. 4 caractères = 3 FRAMES
(1 mot = 3 caractères de 6 bits)

Note.
1600 frames (ou bytes) pàr inch correspondent à 640 frames par centimètre.

Calcul de la capacité réelle d'une bande UNISERVO 12.

$$C_r = \frac{L * F * B}{\frac{F * B + 82}{D} + E} \qquad (1)$$

- L = longueur de la bande (en centimètres)
- F = facteur de blocage
- B = nombre de caractères dans 1 enregistrement logique.
  Considérons F * B = 1-K (mot CPU)
- il y a 41 caractères de synchronisation avant et après les blocs physiques (⟹ 82)
- D = densité de la bande
- E = espace inter-bloc.

on en déduit :

. $C_r = \frac{73.000 * 2.250}{\frac{2.250 + 82}{640} + 1,5} = \pm$ 32 millions de caractères.

(remarque : 1-K (mot CPU) = 3000 caractères CPU; transformé en frames, on obtient :

$\frac{3000 * 3}{4}$ = 2.250 frames ou bytes.)

. la capacité réelle d'une bande dépend de la taille d'un bloc physique.

---

(1) formule tirée de "Les Fichiers" de Claude JOUFFROY et Charles LETANG.

Autres caractéristiques des bandes.

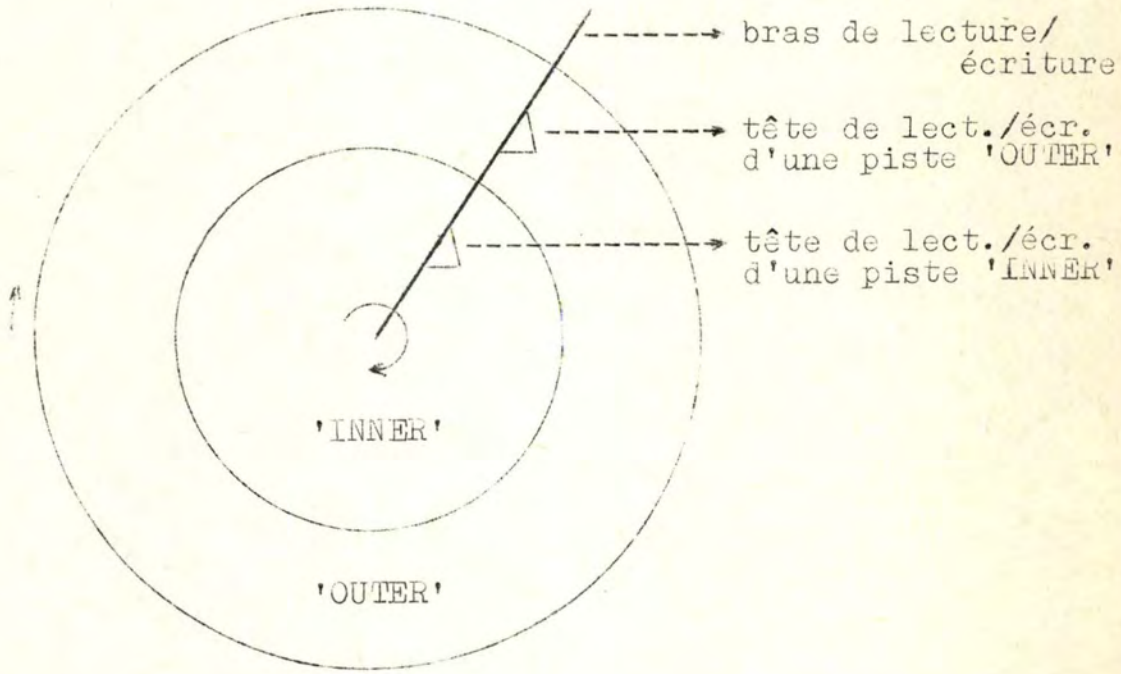
- la lecture ou l'écriture d'une bande prend environ 7 minutes (par tests)
- le sous-système "bande" emploie le procédé du "dual access".

3.1.2. - Les disques.

L'unité périphérique "disque" utilisé est l'UNIVAC '8460'

Caractéristiques.

- l'unité se compose de 2 "positioner modules" (PM) tout à fait indépendants l'un de l'autre.
- chaque PM contient 11 disques (soit 22 surfaces). Seules, 20 surfaces sont utilisées pour stocker les données.
- chaque surface contient 768 pistes dont 384 "INNER" et 384 "OUTER".



- une piste "OUTER" contient 116 secteurs de 56 mots.
  une piste "INNER" contient 92 secteurs de 56 mots.

. par position (1 cylindre "INNER" et 1 cylindre "OUTER), on a accès à (116 + 92) x 20 = 4.160 secteurs. Mais seuls, 4.096 secteurs sont utilisés à stocker les données.

Autres caractéristiques.

capacité d'un PM	88.080.384 mots de 18 bits
capacité d'un UNIVAC '8460'	176.160.768 mots de 18 bits
vitesse de rotation	1.160 tours par minute (± 50 millisecondes par rotation)

Les unités "disque" n'utilisent pas le "dual access".

Section 2. - Procédure par fragmentation du fichier.

3.2.1. - Estimation du fichier.

. Le volume du fichier est estimé à 3,5 milliards de caractères. Les solutions proposées, dans la suite, sont basées sur cette estimation.

. Selon cette estimation, il faudrait environ 110 bandes (enregistrement de 1-K) pour dumper tout le fichier.

. De même, il faudra disposer de 7 unités de disques UNIVAC '8460' pour contenir le fichier.

3.2.2. - Procédure.

(voir fig. 3.2.2.a.)

. Le sous-système "disque" n'utilise pas le "dual access"; il sera donc impossible de vider les deux PM's simultanément.

. Le vidage des PM's s'effectue vers un sous-système "bande" composé d'au moins deux dérouleurs de bandes. La technique du "SWAPPING" sera utilisée. Cette technique fait intervenir le travail de l'opérateur et consiste à aiguiller

. par position (1 cylindre "INNER" et 1 cylindre "OUTER), on a accès à (116 + 92) x 20 = 4.160 secteurs. Mais seuls 4.096 secteurs sont utilisés à stocker les données.

Autres caractéristiques.

capacité d'un PM	88.080.384 mots de 18 bits
capacité d'un UNIVAC '8460'	176.160.768 mots de 18 bits
vitesse de rotation	1.160 tours par minute (± 50 millisecondes par rotation)

Les unités "disque" n'utilisent pas le "dual access".

Section 2. - Procédure par fragmentation du fichier.

3.2.1. - Estimation du fichier.

. Le volume du fichier est estimé à 3,5 milliards de caractères. Les solutions proposées, dans la suite, sont basées sur cette estimation.

. Selon cette estimation, il faudrait environ 110 bandes (enregistrement de 1-K) pour dumper tout le fichier.

. De même, il faudra disposer de 7 unités de disques UNIVAC '8460' pour contenir le fichier.

3.2.2. - Procédure.

(voir fig. 3.2.2.a.)

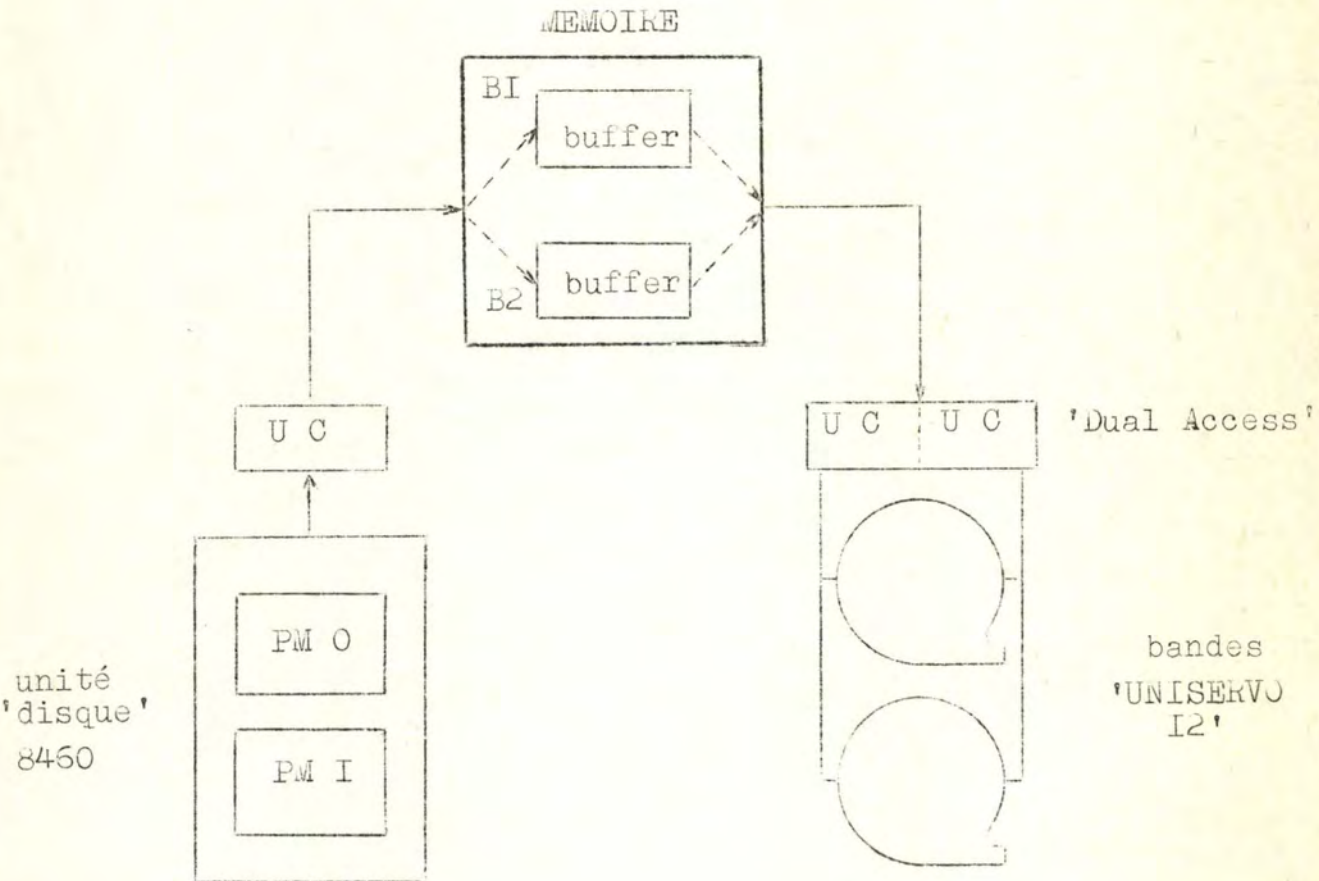
. Le sous-système "disque" n'utilise pas le "dual access"; il sera donc impossible de vider les deux PM's simultanément.

. Le vidage des PM's s'effectue vers un sous-système "bande" composé d'au moins deux dérouleurs de bandes. La technique du "SWAPPING" sera utilisée. Cette technique fait intervenir le travail de l'opérateur et consiste à aiguiller

le flot des données vers la bande "vierge" du second dérouleur dès que celle du premier est remplie, et inversément. L'opérateur se chargera de placer une nouvelle bande "vierge" sur le dérouleur inactif avant la fin du remplissage de la bande se trouvant sur l'autre dérouleur.

. On dispose de deux buffers en mémoire centrale. Leur volume sera établi ultérieurement et dépendra, entre autres choses, de la place mémoire disponible.

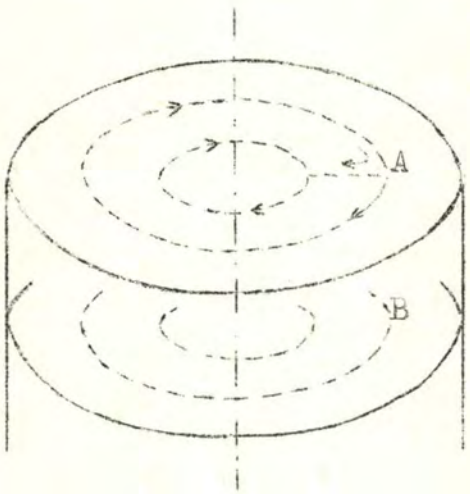
NOTONS également qu'il est possible d'effectuer la même procédure simultanément sur le second ordinateur qui possède sa propre mémoire, identique à celle du premier CPU.



(fig. 3.2.2.a.)

3.2.3. - Vidage d'un "positioner module".

3.2.3.1. - Choix des buffers.



- Un "I/O call" permet de transférer, au maximum et en l'espace de deux rotations de disque, une piste "OUTER" et une piste "INNER".
- Il est possible, pendant le temps de transfert de ces deux pistes (2 rotations = ± 100 millisecondes), de préparer l'I/O suivant de sorte que le temps de passage de A à B se fasse sans perte de temps.
- Pour rappel : il existe une tête de lecture pour la piste "INNER" et une pour la piste "OUTER". De plus, le passage d'une surface à l'autre se fait sans changement de position du bras de lecture, sauf lorsqu'il est nécessaire de changer de cylindre, c'est-à-dire tous les 20 I/O's (il y a 20 surfaces utilisées).
- Afin de profiter au maximum des avantages fournis par le '8460' (lecture en 2 rotations d'une piste "OUTER" et d'une piste "INNER"), il est nécessaire de disposer de buffers de 11.648 mots. Cette longueur provient du fait qu'une piste "INNER" contient 5.152 mots (92 secteurs * 56 mots) et qu'une piste "OUTER" contient 6.496 mots (116 secteurs * 56 mots).

. Le problème devient :

> les bandes sont-elles capables d'absorber le contenu d'un buffer de 11.648 mots avant même que le second buffer ne soit rempli à partir du disque ?

Notes.

1. Le temps de transfert du disque vers le buffer est égal à la durée de 2 rotations "disque" (donc ± 100 millisec.)
2. La place disponible en mémoire centrale (± 25-K) permet uniquement l'utilisation de deux buffers de 11.648 mots. Une procédure à quatre buffers serait possible si la place mémoire le permettait; dans ce cas, le volume des données transférées, dans la même unité de temps, serait doublé. (Cette solution pourrait s'appliquer dans le cas d'un dump "off-line").

3.2.3.2. - Calcul du temps théorique de vidage d'un PM.

. La réponse au problème posé se fera par test.

. Enoncé du test :

> dump de 128 cylindres (64 "INNER" et 64 "OUTER") selon le procédé décrit précédemment.

. Le temps d'exécution de ce dump varie autour de 210 secondes. Or, un "positioner module" contient 768 cylindres (384 "INNER" et 384 "OUTER"), c'est-à-dire six fois plus que dans le test. On peut donc logiquement conclure que le temps théorique pour le vidage d'un PM est 21 minutes.

Remarque.

L'emploi d'une procédure identique par le second ordinateur doublerait le volume des informations transférées pendant la même période.

3.2.3.3. - Taux de transfert des bandes.

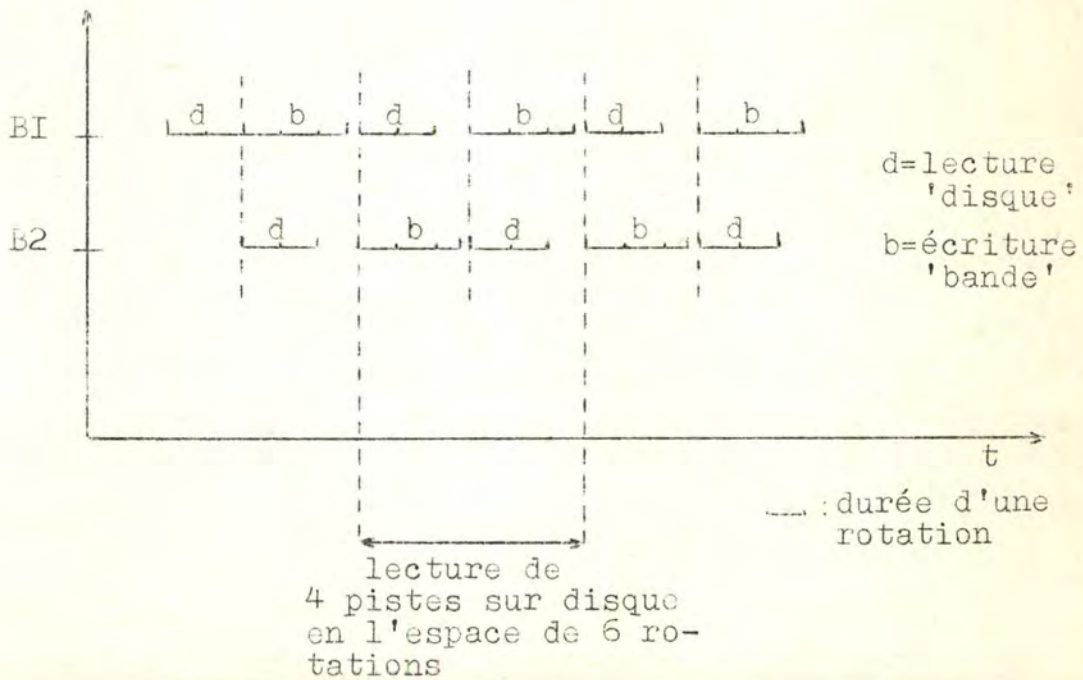
- Un I/O disque (c.à.d. la lecture d'une piste "INNER" et d'une piste "OUTER") exige la durée de deux rotations. Dans le cas le plus favorable, la lecture de 128 cylindres (TEST) exigerait 2.560 rotations.
  En effet, il existe 20 surfaces utilisables. De plus, par surface, 2 rotations sont nécessaires pour lire la piste "INNER" et la piste "OUTER". Enfin, le test s'effectue sur 64 positions (de bras).

- Le test a exigé 210 secondes, à 50 millisecondes la rotation. Il a donc fallu 4.200 rotations pour effectuer le dump.

- Nombre de rotations perdues :
  4.200 - 2.560 = 1.640 rotations
  ce qui constitue 39 % du total des rotations.

- On peut donc considérer que l'on lit approximativement deux pistes (1 "INNER" et 1 "OUTER") en trois rotations, la différence provenant notamment des changements de position du bras (1 par cylindre) et de l'existence de "Spares". Les "Spares" sont des secteurs spéciaux (32 par cylindre); ils constituent une réserve de secteurs, disponibles dans le cas où certains secteurs "normaux" seraient détruits.

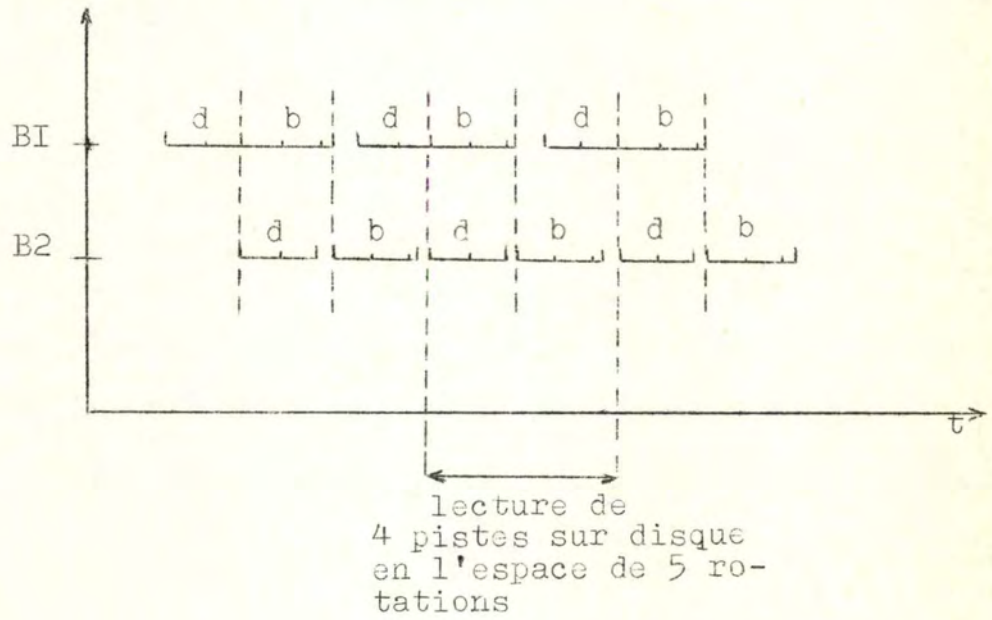
- Etablissons maintenant un schéma d'utilisation des buffers.



. Le schéma ne tient pas compte des mouvements du bras de lecture et de l'existence des "Spares".

. L'utilisation des buffers telle qu'elle est présentée, se base sur un temps de lecture équivalent à la durée de trois rotations pour deux pistes, tel qu'il a été établi précédemment.

. On peut, dès lors, en conclure que la vitesse de transfert du buffer vers les bandes est comprise entre 125 et 150 millisecondes. Une vitesse de transfert inférieure à 125 millisecondes rendrait possible le schéma suivant :



. Un tel schéma aurait nécessité un temps d'exécution du test nettement inférieur aux 210 secondes observées.

. La vitesse des bandes peut, dès lors, être estimée à 85.000 mots par seconde.

3.2.4. - Organisation du sauvetage du fichier.

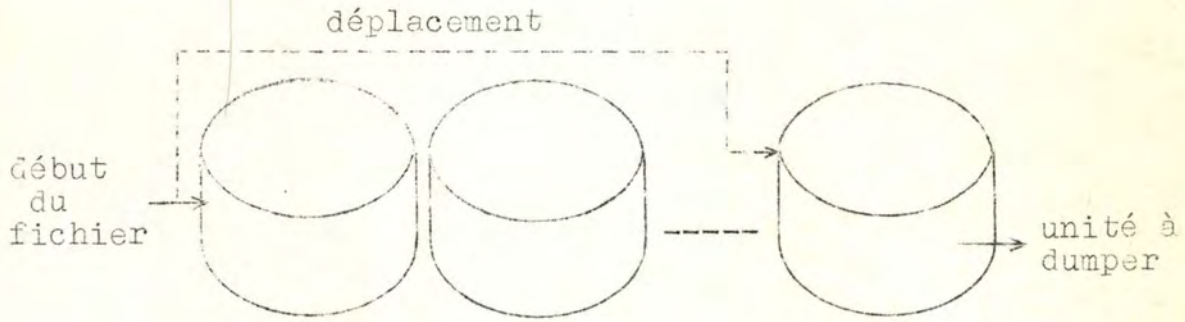
L'estimation du fichier (± 3,5 milliards de caractères) nous impose de disposer de 7 unités "disque" 8460 (+ 1 unité de réserve).

Il a été défini également que le temps théorique de vidage d'un PM nécessiterait 21 minutes. Il est donc possible, en l'espace d'une heure et en utilisant les 2 ordinateurs, de permettre le vidage de deux unités '8460'.

Le sauvetage du fichier s'effectuerait cycliquement tous les 4 jours, à raison d'un "dump" journalier de 1 heure. Il n'est pas encore tenu compte, à ce stade, du problème constitué par la gestion des "afterlooks", étant donné la fragmentation du fichier.

Remarque.

1. Sur disque, de même que sur tambour ou sur bande, un I/O est toujours "logique". Il est difficile de demander le "dump" d'une unité (physique) comme le '8460'.
   En fait, il suffirait de connaître l'adresse physique d'implémentation du fichier (c'est possible), de même que le déplacement par rapport à cette adresse, de début de fichier.



2. Problème de sécurité concernant l'organisation du fichier. - Le problème concerne la répartition des "buckets" sur les 7 unités "disque 8460".

Ière solution : implémentation des "buckets" sur une seule unité.

avantage : facilité d'exploitation.

désavantage : si cette unité tombe en panne, tout le fichier est inexploitable.

2me solution : répartition des "buckets" sur les sept unités '8460'.

avantage : si une unité tombe en panne, les six autres sont toujours exploitables.

désavantage : complexité d'exploitation.

3me solution : double copie de tous les buckets sur deux unités différentes.

avantage : degré plus élevé de sécurité.

désavantage : la mise à jour des "buckets" doit se faire à deux reprises.

3.2.5. - Administration des "afterlooks".

La segmentation finale du fichier dépendra également du travail que nécessitera la gestion journalière des "afterlooks".

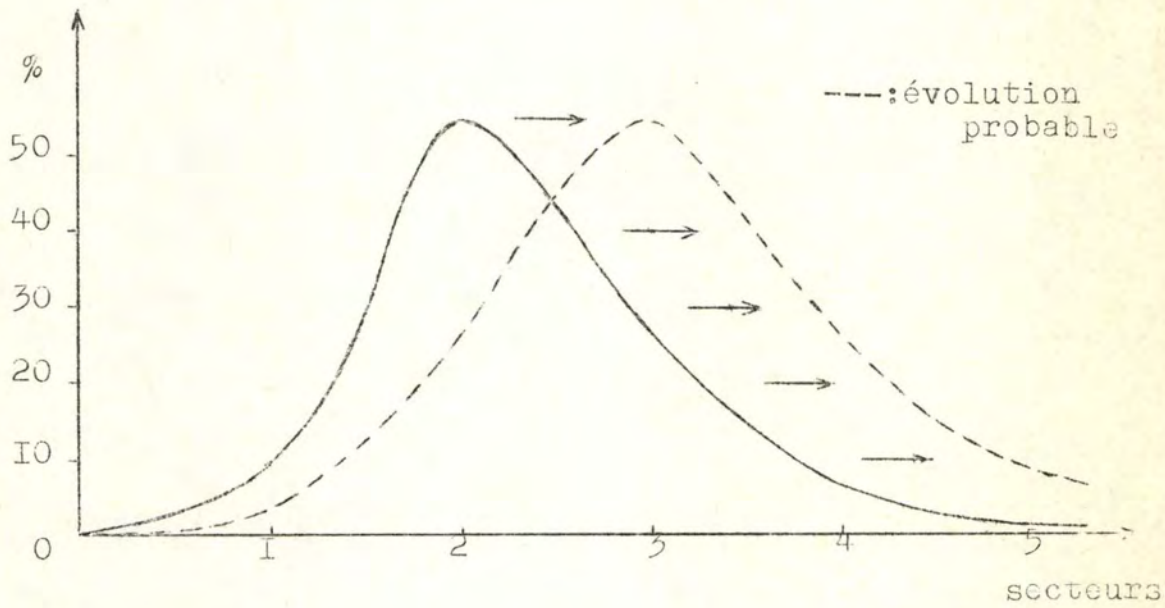
Si l'on considère un nombre moyen de transactions à 200.000 par jour, on peut estimer, au même nombre, le total journalier des "afterlooks".

3.2.5.1. - Estimation du nombre de bandes nécessaires à l'enregistrement des "afterlooks" d'une journée.

A. Le nombre de bandes utiles est dépendant de la longueur des "afterlooks". Un test effectué sur 246 transactions donne les résultats suivants :

Nombre d'afterlooks	Longueur en secteurs	Pourcentage
23	1	9,5
136	2	55,9
67	3	27,7
15	4	6,-
1	5	0,9

Distribution de la longueur des enregistrements AFL's.



Dans l'avenir, il est prévu d'ajouter certaines informations dans les enregistrements; de plus, de nouvelles applications seront introduites et contribueront également à allonger la longueur moyenne des enregistrements. Celle-ci tendra vers 3 secteurs (un secteur = 56 mots).

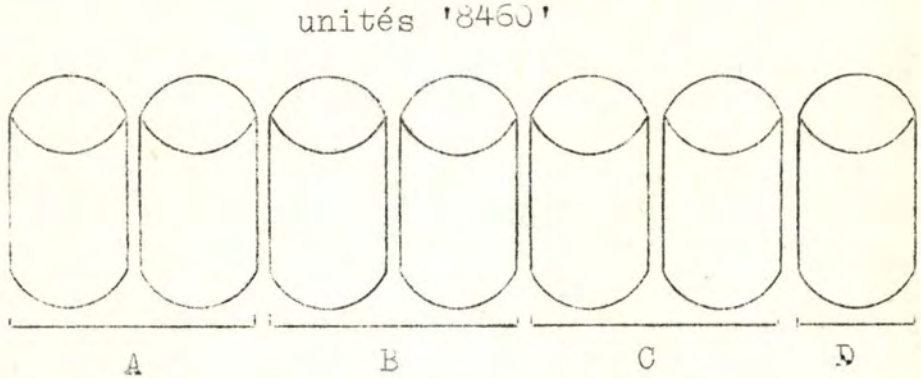
B. Rappel.

Estimation du nombre de transactions : 200.000/jour
Longueur moyenne d'un AFL. : 3 secteurs (± 150 mots)

Le volume journalier des "afterlooks" sera d'environ 30 millions de mots, ce qui correspond ± au volume de 3 bandes.

3.2.5.2. - Procédure de gestion des "afterlooks".

. Supposons que le fichier soit divisé en 4 groupes :



. Un de ces groupes sera dumpté chaque jour.

. La segmentation du fichier en quatre groupes exige un travail supplémentaire en fin d'exploitation journalière. Il s'agit de répartir les "afterlooks" de la journée sur des bandes propres à chacun des groupes.
Si l'on effectue, après la fin de l'exploitation du temps-réel, un dump du groupe A, il sera utile également d'exécuter un "splitsing" des bandes AFL de la journée vers les bandes propres aux groupes B, C et D, les "afterlooks" de A n'étant plus nécessaires.
Cette gestion journalière des "afterlooks" est représentée par la figure 3.2.5.2.a.

. Lorsqu'un groupe a fait l'objet d'un dump, toutes les bandes AFL's, propres à ce groupe et découlant de la gestion des journées précédentes, sont également supprimées.

. La figure 3.2.5.2.b. représente le volume des AFL's à conserver au fil des jours.

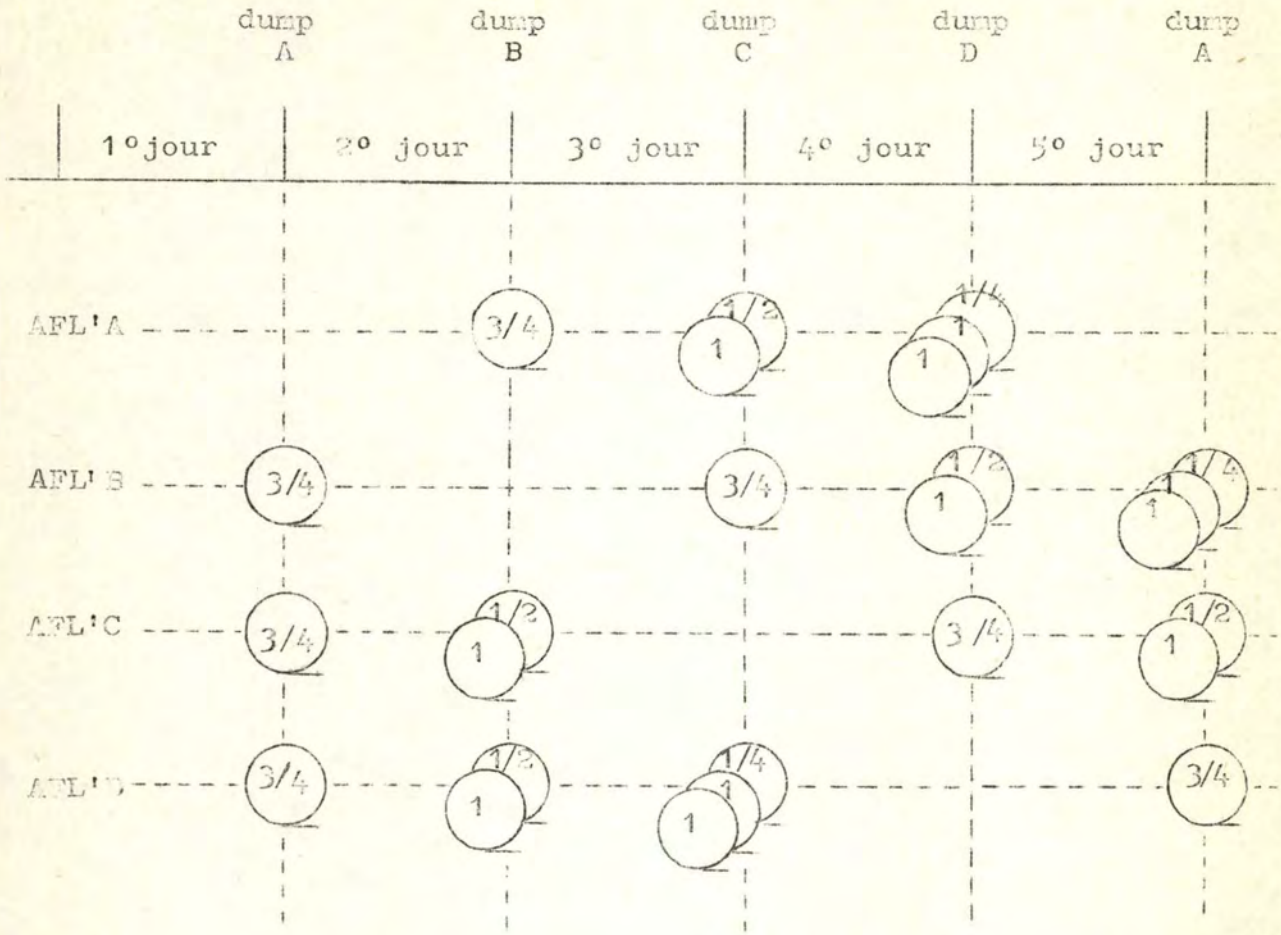


fig. 3.2.5.2.b. - Volume des "afterlooks"

Commentaires.

. Un dump du groupe A est effectué en fin d'exploitation de la première journée. Nous avons établi que le volume journalier des "afterlooks" correspondait approximativement au volume de 3 bandes.

On exécute, dès lors, et de la façon décrite à la fig. 3.2.5.2.a. un "splitsing" de façon à répartir les "afterlooks" des groupes B, C et D vers les bandes propres à ces groupes. En fait, étant donné que les "afterlooks" de la journée, concernant le groupe A, sont éliminés, une bande remplie approximativement aux 3/4 sera suffisante pour emmagasiner les AFL's des groupes B, C et D.

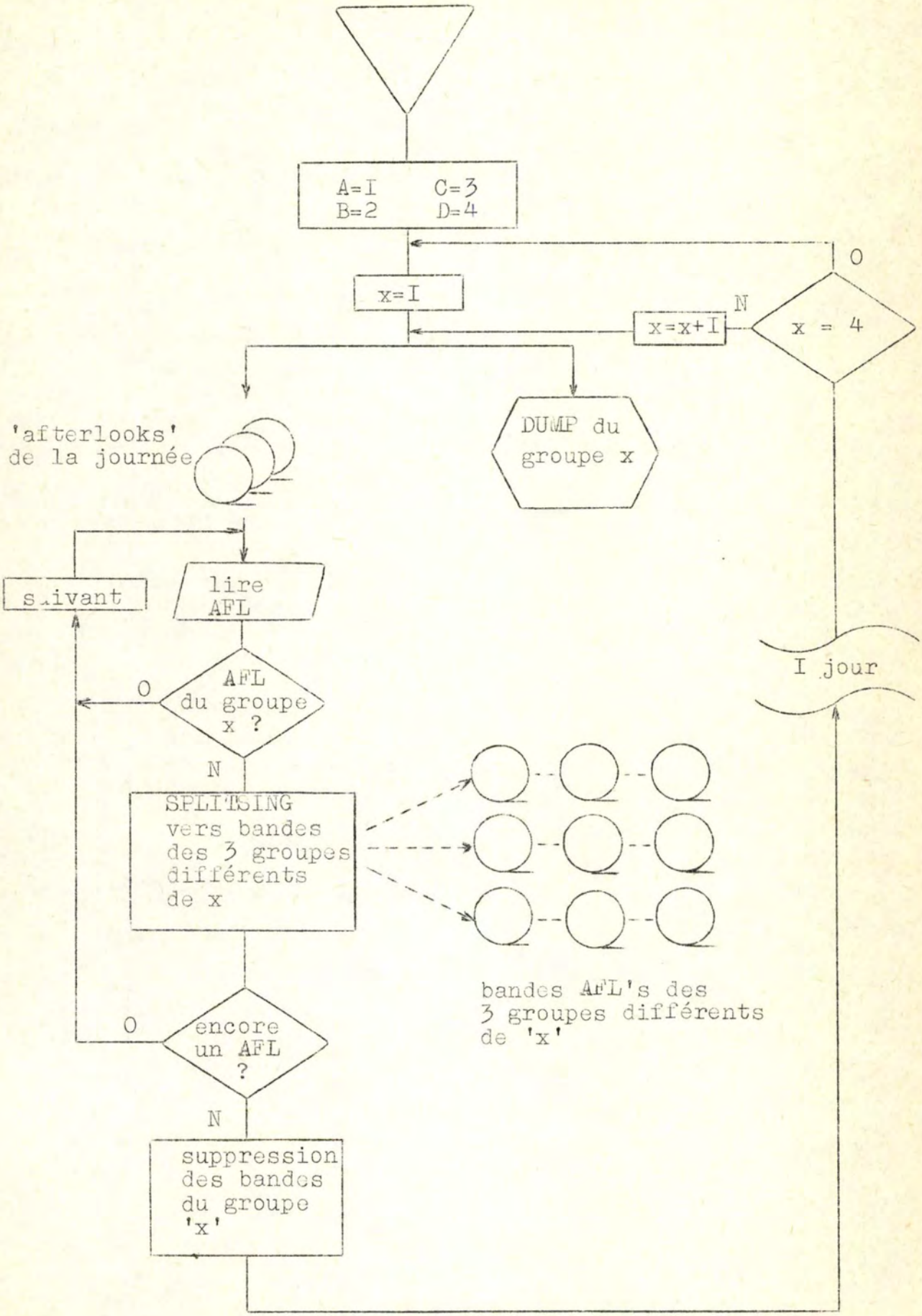


fig.3.2.5.2.a. Gestion journalière des 'afterlooks'

. En fin d'exploitation du jour suivant, sera effectué un dump du groupe B. Le principe reste identique et, tandis qu'il n'y aura plus d'AFL's du groupe B à conserver, les AFL's pour les autres groupes s'ajouteront à ceux des journées précédentes.

. Il en va de même pour toutes les journées. On constate que le maximum d' "afterlooks" à conserver pour un seul groupe sera contenu dans trois bandes.

3.2.6. - Découpe optimale du fichier.

La découpe du fichier doit être fixée en fonction de deux critères essentiels :

1. la durée du travail à exécuter en dehors des heures d'exploitation du temps-réel. Ce travail comprend :
   - l'exécution du dump
   - le "splitsing" des "afterlooks"
2. le temps nécessaire à un éventuel recouvrement du fichier. Plus la découpe est fine , plus le recouvrement se fera rapidement.

Il y a un équilibre à trouver entre ces deux critères, ceux-ci étant complémentaires l'un par rapport à l'autre.

Une amélioration de l'un signifiera une dépréciation de l'autre.

La procédure de choix utilisée sera la suivante :

1. établissement des courbes de durée du dump et du "splitsing".
2. établissement de la courbe de durée totale (somme des deux courbes précédentes).
3. la découpe choisie répond-elle au second critère du choix ?

La figure 3.2.6. nous donne une allure approximative des courbes de durée.

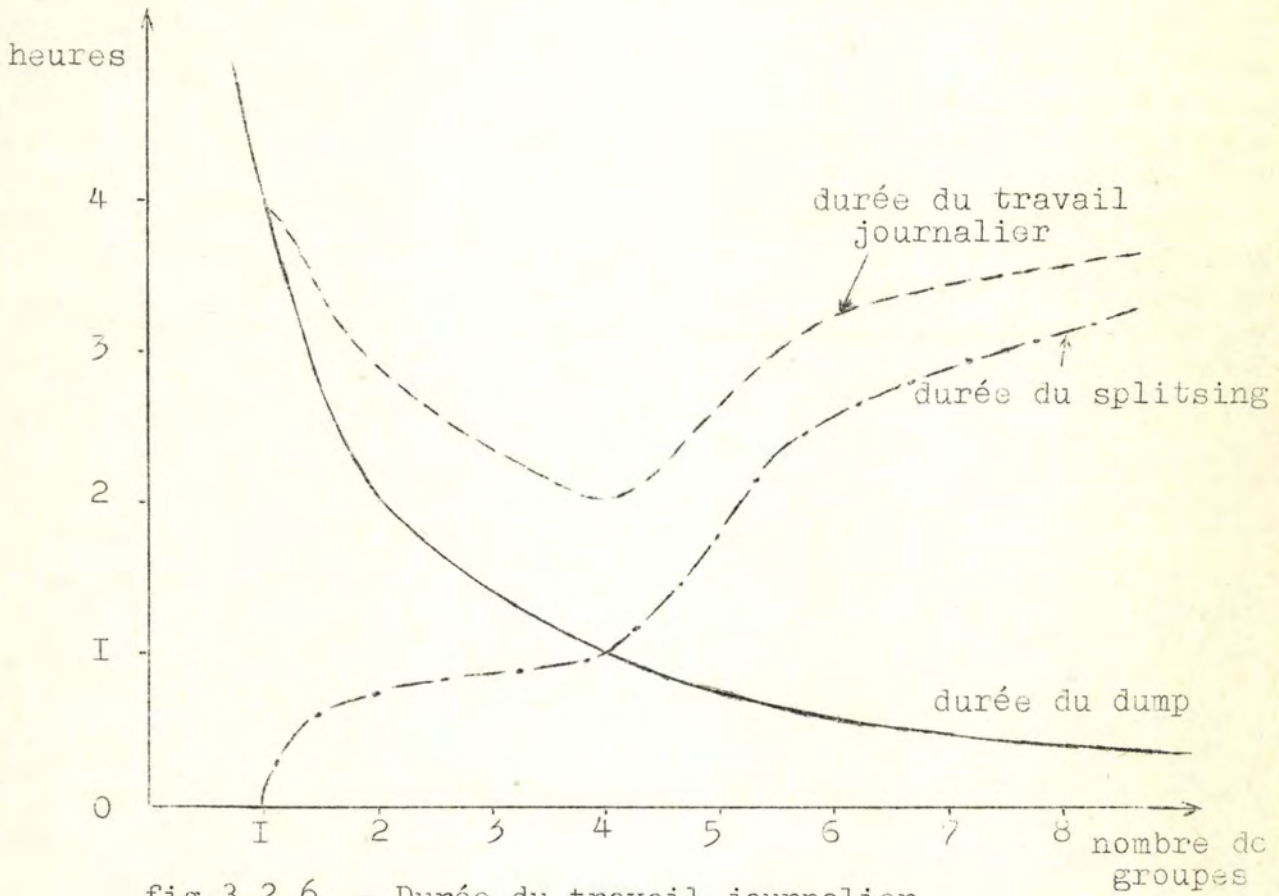


fig.3.2.6. - Durée du travail journalier en fonction de la segmentation du fichier.

- Courbe de durée du dump.

  Cette courbe tient compte de l'emploi simultané de deux procédures équivalentes sur les deux ordinateurs.

- Courbe de durée du "splitsing".

  Le volume journalier des 'afterlooks" est invariant et se situe aux alentours de 30 millions de mots (± 3 bandes). Par contre, plus le nombre de groupes augmente et plus le nombre de dérouleurs de bandes nécessaires au "splitsing" augmente également. Si le nombre de dérouleurs de bandes (9 tracks) est limité à 4 par sous-système, une découpe

du fichier en plus de 4 groupes, doublerait quasiment la durée du "splitsing".

On peut considérer qu'un "splitsing" à partir des trois bandes AFL's de la journée vers les bandes des différents groupes (si le nombre de ces groupes est inférieur ou égal à 4) peut se réaliser dans un intervalle d'une heure.

(Note : le "splitsing" s'exécute sur un seul ordinateur.)

- La découpe en quatre groupes semble offrir un travail quotidien minimum (± 2 heures).
- La troisième étape de la procédure de choix est du ressort du chef de centre. C'est à lui de décider s'il est prêt à interrompre l'exploitation pendant une durée plus ou moins grande, en cas de reconstruction du fichier.

Remarque.

L'apport de nouvelles unités UNISERVO 12 tendrait à favoriser une découpe plus fine du fichier.

Section 3. - Les procédures "ON-LINE".

Le Projet prévoit, pour l'avenir, une exploitation continue du système (24 heures sur 24). Une procédure "ON-LINE", c'est-à-dire pendant l'exploitation du temps-réel, s'avèrera dès lors indispensable.

Le dump journalier se baserait toujours sur une segmentation du fichier; il pourrait s'exécuter de deux manières différentes, suivant qu'il y ait réservation ou non de la partie du fichier à dumper.

3.3.1. - Dump sans réservation du fichier.

Exécuter un dump sans réserver, préalablement, la partie du fichier à dumper, signifie que l'exécution de

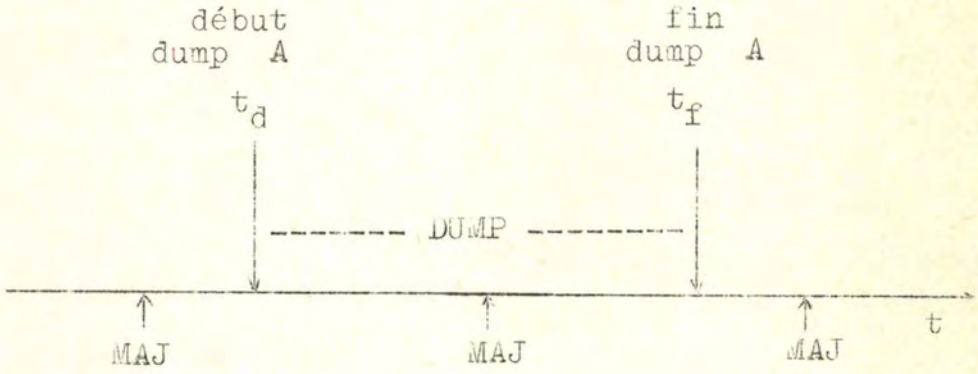
ce dump pourra être interrompue par les transactions du temps-réel qui sont prioritaires par rapport au programme de dump.

Cela signifiera également qu'un enregistrement de la partie à dumper pourra être modifié pendant l'exécution même du dump.

Il s'ensuit que ce dump ne représentera jamais l'image du fichier à un instant donné de l'exploitation et que l'administration des "afterlooks" en sera d'autant plus compliquée.

3.3.1.1.- Schéma d'exécution.

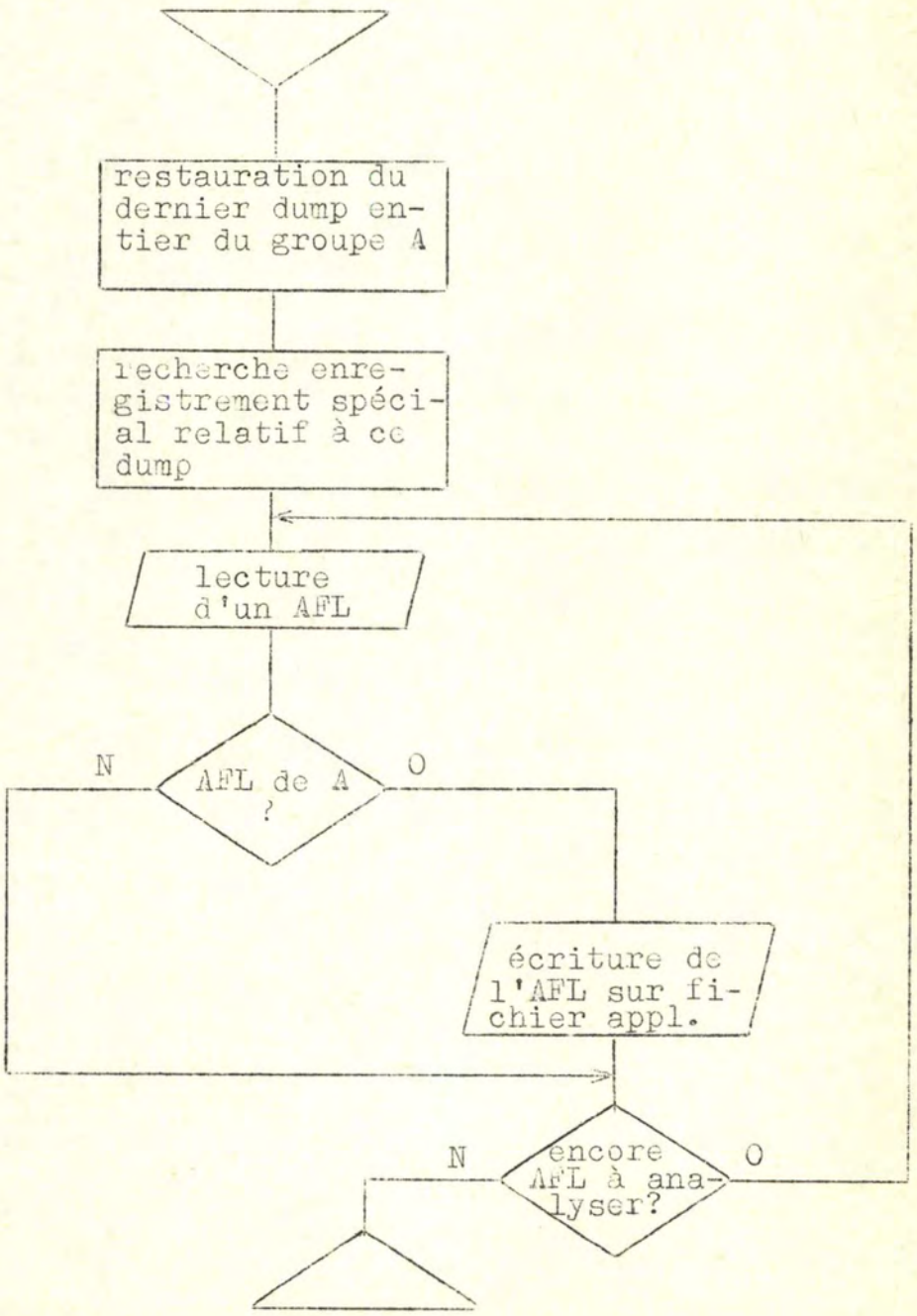
Afin de faciliter l'analyse, nous supposerons que le fichier sera décomposé en deux parties (A et B) et qu'une de ces parties fera l'objet d'un dump journalier.



MAJ : mise à jour d'un enregistrement de la partie A

- Le dump s'exécutera à une période creuse de l'exploitation; de cette façon, il exigera un temps d'exécution minimum, les interruptions en provenance des transactions y étant peu fréquentes.
- Concernant la gestion des "afterlooks", un enregistrement spécial sera ajouté au fichier AFL au début de chaque dump (en t_D). Ainsi donc, s'il intervenait un incident

après t_f nécessitant le recouvrement de la partie A du fichier, il suffirait de restaurer le dump pris entre t_D et t_f et d'y ajouter tous les "afterlooks" de la partie A à partir de l'enregistrement spécial introduit dans le fichier des AFL's en t_D.
Si l'erreur s'était produite entre t_D et t_f ou avant t_D, le processus eût été le même mais en utilisant le dump précédent de A (ancien de deux jours) et en considérant l'enregistrement spécial relatif à ce dump dans le fichier des "afterlooks".



Le schéma précédent décrit la procédure à exécuter en cas de recouvrement d'une partie du fichier (soit du groupe A).

3.3.1.2. - Remarques.

1. Les "afterlooks" peuvent être contenus dans plusieurs bandes.

2. L'enregistrement "spécial" doit pouvoir identifier, de façon unique, le dump pour lequel il a été créé.

3. Le test déterminant si l' "afterlook" lu concerne le groupe A consiste en une comparaison d'adresses se trouvant dans les enregistrements "afterlooks". Il s'agit de l'adresse de l'enregistrement du fichier pour lequel cet "afterlook" a été pris.

4. Notons que l'on n'effectue aucun "splitsing"; le temps d'exécution d'une telle procédure ne le permettrait pas.
   Les "afterlooks" de toutes les parties du fichier sont donc conservés, ensemble, sur les mêmes bandes.

3.3.2. - Dump avec réservation du fichier.

Cette procédure utilise le principe de la "lock-list". Elle se réserve, pour la durée de son exécution, la partie qu'elle se propose de dumper.

La procédure est identique à celle d'un dump "on-line" sans réservation sauf qu'il en résulte un dump qui sera l'image exacte de la partie du fichier pour laquelle il est exécuté.

Cette solution possède, néanmoins, de sérieux inconvénients :

- l'utilisation de la "lock-list" signifie que la réservation se fera au niveau des "buckets". De ce fait, le dump risque d'exiger un temps d'exécution sensiblement plus long, le dump se faisant enregistrement par enregistrement.

- le fait de réserver la partie du fichier à dumper aura une influence défavorable sur le temps de réponse de certaines transactions.

==========

Cette analyse ne prétend pas apporter une solution définitive au problème qu'il évoque. Elle souligne, avant tout, les difficultés et espère, de ce fait, avoir stimulé la réflexion et la discussion des membres du Projet chargés de mettre en oeuvre le système.

==========

C O N C L U S I O N S .

On ne peut établir de règles formelles ni précises quant aux protections à exercer dans tel ou tel domaine. Il n'existe pas de bréviaire en cette matière. Ce qui importe, c'est l'état d'esprit qui préside aux choix des voies suivies. La mise en oeuvre de mesures de protection est donc plus le fruit d'un jugement que l'application de normes toutes faites. Notons cependant quelques suggestions intéressantes pour une meilleure sécurité dans un système en temps-réel. Ces suggestions ne représentent, en aucun cas, une "checklist". Elles sont basées sur des observations et des commentaires de personnes ayant une certaine expérience dans ce domaine.

A. - Sécurité "physique"

1. Contrôle soigneux de toutes les unités physiques du système.

   Certaines unités, telles que le conditionnement d'air et les unités périphériques requièrent une attention toute particulière.

2. Mise en oeuvre de procédures d'alarme.

   En cas d'erreur grave dans le déroulement normal des opérations ou de mauvais fonctionnement d'une unité physique, certains dispositifs seront prévus :

   - sonnettes d'alarme
   - voyants lumineux
   - messages "accrocheurs" à la console

3. Limitation des accès à la librairie des bandes.

   Il est commun de voir de nombreuses personnes accéder à la librairie des bandes. Il importe que ces accès soient contrôlés de façon stricte.

4. Contrôle de l'accès aux terminaux.
Tout terminal pouvant être utilisé pour accéder aux données du système, sera "verrouillé" ou placé dans un endroit protégé.

5. Dédoublement des unités.
Le dédoublement de l'unité centrale constitue une protection indispensable dans un système en "temps-réel". Le dédoublement d'autres unités (périphériques, contrôleurs, ...) dépendra généralement d'une analyse critique du coût de la mesure et du prix d'un risque calculé.

B. - Sécurité "logique"

1. Etablissement de procédures, d'instructions pour les opérateurs.
Il est difficile de se protéger contre le mauvais jugement ou l'incompétence des individus. Une "check list" à l'intention des opérateurs, contribuera à un degré plus élevé de sécurité. Insistons également sur l'importance d'une formation à la fois technique et générale du personnel d'un centre de traitement de l'information.

2. Importance des tests.
Un programme insuffisamment testé peut avoir des répercussions inattendues et désastreuses (erreurs en chaîne). La responsabilité du programmeur est mise en question.

3. Contrôle périodique de la validité et de la vraisemblance des données dans les fichiers "application".
L'intégrité des informations est un souci primordial en "temps-réel". Le contrôle de validité et de vraisemblance contribue à ce souci.

4. Importance capitale des procédures de secours.
Il importe que tous les types d'erreurs et de mauvais fonctionnement d'un composant du système soient pris

en considération par ces procédures. Celles-ci reposent, entre autres, sur la conservation adéquate des fichiers "critiques".

5. Vérification des "labels".
Cette mesure est indispensable dans un système en "temps-réel". Son oubli peut entraîner des conséquences désastreuses dans le cas de bandes supportant les fichiers "critiques".

.....

Il nous a semblé que certains principes méthodologiques devaient prévaloir dans la mise en oeuvre de ces mesures de protection. S'ils ne sont pas spécifiques au "temps-réel", ils acquièrent, dans un tel système, un poids plus important encore.

Nous dégageons quatre principes fondamentaux :

Premier principe : "Refus des solutions de facilité."
Ces solutions s'accompagnent généralement d'une perte dans l'efficacité du système. C'est ainsi qu'un "COLD RESTART" est une solution de facilité, entraînant une perte d'informations.

Deuxième principe : "Définition des responsabilités."
Des responsabilités spécifiques pour les différentes tâches seront établies. Les opérations de conception, d'écriture, d'exécution, ... des programmes seront assignées à des personnes différentes. La séparation effective des différentes fonctions aura pour effet de réduire les risques d'erreurs et les actes délibérés non autorisés.

Troisième principe : "Concerne le coût de la sécurité."
L'utilisateur doit garder une juste mesure entre le coût des mesures de protection, qui assurent un degré de plus en plus élevé de sécurité, et le prix d'un risque calculé.

On a souvent tendance à user de la facilité des tests automatiques sans trop réfléchir au prix que l'on paie pour atteindre l'objectif que l'on se fixe.

Quatrième principe : "La sécurité fait partie intégrante du traitement de l'information".

Ce principe nous ramène à une idée déjà maintes fois exprimée : on doit être convaincu que la sécurité consiste en un ensemble de mesures que l'on adopte dès que l'on décide d'utiliser un ordinateur.

Notons également que la fonction de sécurité ne limite pas son intervention à la protection des informations et à la continuité des opérations. Elle s'étend, en fait, à toutes les activités d'un centre de traitement.

.....

Les principes énoncés montrent combien le personnel d'un centre de traitement demeure un souci constant à la fois au titre du groupe dans lequel il s'intègre et à titre individuel.

Le "temps-réel" valorise, plus que n'importe quel autre type de traitement classique de l'information, le travail de groupe. Il nous semble donc que l'individu mérite, à cet égard, des soins attentifs; il requiert une préoccupation permanente de formation technique et de culture générale.

Le "temps-réel" est, avant tout, un "travail en communauté".

==========

G L O S S A I R E .

Afin d'éviter toute confusion, une même terminologie a été conservée d'un bout à l'autre de l'analyse. Les termes techniques définis sont, dans la plupart des cas, repris de la terminologie propre à UNIVAC.

"ABORT"

Terminaison anormale d'une activité d'une transaction due à une erreur qui a été détectée.

Activité

Au niveau du RTOS : séquence d'instructions logiquement ininterruptible.
Au niveau du RTS : appel pour une fonction.

"activity packet"

Bloc de contrôle représentant une demande pour une page. Physiquement, il s'agit d'un enregistrement de 32 mots comprenant l'adresse virtuelle de la page, les limites de la TAR de la transaction qui demande l'exécution de cette page et le contenu des registres du système.

"afterlook"

Image d'une portion de fichier après qu'elle ait été mise à jour.

"bay"

Segment de 4 K en mémoire centrale.

"CRITICAL AREA"

Informations concernant la configuration du système, à un moment donné.

DBJ (data base job)

Programme "batch" s'exécutant sous le run RTS et bénéficiant des facilités de ce run (fonctions "SYSTEME", procédures de recovery, ...)

"Duplex files"

Certains fichiers critiques du système sont conservés en deux copies. Exemples de "duplex files" : fichier des afterlooks, fichier des beforelooks, fichier FH-TAR, ...

FAR (fil area)

Zone de mémoire principale allouée à une transaction. Cette zone a, pour objectif, de recevoir l'enregistrement du fichier "APPLICATION" que la transaction se propose de modifier.

FH-TAR

Image sur tambour, de la TAR. Il existe une zone FH-TAR allouée à chaque terminal.

Fichiers "APPLICATION"

Fichiers contenant les données de l'utilisateur, par opposition aux fichiers "SYSTEME".

Fonction

Suite d'instructions dont le résultat est un changement de certaines données ou de l'état du système.

Interaction

Partie du traitement d'une transaction contenue entre deux interventions successives de l'utilisateur au terminal.

"lock list"

Moyen permettant à une interaction d'interdire l'accès à une portion de fichier, par toutes autres interactions.

"logging"

Certaines données du temps-réel sont conservées sur bande pour être l'objet d'un traitement "batch" ultérieur.
La bande "logging" contient également des informations concernant des erreurs survenues ("dump" de l'environnement d'une transaction, par exemple).

Multifil ("multithreading")

Plusieurs messages sont traités simultanément en parallèle. Différents programmes travaillant à des messages différents sont tous à des stades variés d'exécution simultanée.

Page

Unité de stockage de l'information (1-K).
Il y a des pages virtuelles (sur tambour) et des pages "hardware" (en mémoire principale) destinées à recevoir les pages virtuelles.

Ré-entrance

Caractéristique d'une page. Une page est ré-entrante si elle ne se modifie pas elle-même, de sorte que plusieurs transactions concurrentes peuvent exécuter cette page simultanément.

RTOS ("Real-Time Operating System")

"Software" standard de l'UNIVAC 418-III.

RTS ("Real-Time System")

Appellation du système multifil de contrôle des transactions.

Run

Un run consiste en une série de jobs à exécuter pour un seul utilisateur. Les jobs d'un même run sont exécutés en séquence mais différents runs peuvent se dérouler en parallèle.

"run library"

Une librairie du run est allouée par le système dès qu'un run est initialisé. Cette librairie sera conservée sur tambour pendant toute la durée du run. Cette zone servira à garder les différents éléments de programmes nécessaires à l'exécution des différents jobs du run.

Séquenceur

Distributeur des fonctions "SYSTEME".

"SWITCHER"

Distributeur des fonctions "PROCESSEUR" et des TPS's.

TAR ("Transaction Area")

Zone mémoire de 1-K réservée au traitement d'un message.

TCS ("Transaction Control System")

Appellation du système monofil de contrôle des transactions.

Transaction

Entité logique représentant la demande faite par un utilisateur à un moment donné. Une transaction peut se composer d'une ou de plusieurs interactions.

TPS ("Transaction Processing Segment")

Programme "application" de traitement des transactions.

==========

B I B L I O G R A P H I E

LIVRES.

"Design of on-line computers systems" YOURDON

"Design of Real-time systems" J. MARTIN

"Utilisation et programmation des systèmes en temps-réel." J. MARTIN

"File security in on-line computer system" dans REAL TIME (international computer state of the art report) STRANACK

"Guide to the design of real-time system" M.F.ROTHSTEIN

"Security, accuracy and privacy in computer systems". James MARTIN

"Computer security management" Dennis Van TASSEL

"Gestion en temps-réel" (Centi - 3 S)

"Les Fichiers - Pratique et choix de l'organisation de données informatiques". Claude JOUFFROY et Charles LETANG

"Principes d'action et d'organisation en informatique". Jacques BRUNIAT

"Systèmes en temps-réel" S.de HEPCEE

Documentation "UNIVAC 418-III"

"Gestion d'une bibliothèque de bandes en DOS". (article du 2 avril 1968) A. COUZINIE

ARTICLES.

"On the implementation of security measures in information systems".

{communication of the ACM
{avril 1972, Vol.15, n° 4

"Protection du secret des données".

{l'Informatique - janvier 1973.
{M.G. GIRSDANSKY

"Vertrouwelijkheid medische gegevens in verband met databanken" J.N.HERBSCHLEB

"Privacy-aspecten van de automatisering van de informatieverwerking" B.K.BRUSSARD

"Organisation d'un centre de programmation"
Daniel COMONT

"La sécurité dans les systèmes informatiques en temps-réel".

{Informatique et gestion
{1970 août/sept. n° 20.

15th GUIDE conférence - Proceedings (June 4-7)

==========

E R R A T A

- p.8 titre "2. HTAB" au lieu de "2. hATB"
- p.I9 3°lig. Lire:"le message se trouve-t-il...."
- p.24 6°point (entre parenthèses) Lire:"Dat-Base Jobs...")
- p.26 I°lig. Lire:"L'enregistrement d'un individu contiendra notamment les modules..."
- p.55 3°lig. dans les réflexions
  Lire:"les points d'entrées spécifiés..."
- p.72 2°lig. Lire:"...celles qui sont en "output-queue"...
- p.83 4°lig. Lire:"Il n'est pas question qu'un programme, en "batch pur", s'exécute sous la même priorité que le RTS."

  3°point - I°lig. Lire:"close-down"
- p.II8 2°lig. Lire:"Un de ces groupes sera dumpé..."